Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Aplicación de MILP para flowshop sin permutación

Autora: Beatriz Pérez Barranco
Tutor: Víctor Fernández-Viagas Escudero

Dpto. Organización Industrial y Gestión de Empresas
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Aplicación de MILP para flowshop sin permutación

Autora:
Beatriz Pérez Barranco

Tutor:
Víctor Fernández-Viagas Escudero

Dpto. de Organización Industrial y Gestión de Empresas
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Trabajo Fin de grado: Aplicación de MILP para flowshop sin permutación

Autora: Beatriz Pérez Barranco

Tutor: Víctor Fernández-Viagas Escudero

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

Agradecimientos

A mi familia, que me ha apoyado incondicionalmente. Mis padres y mi hermano, que confían en mí más que yo misma y sin los que no habría llegado a dónde estoy ahora.

A mis amigos, que siguen ahí después de tantos años y altibajos que han sufrido como si fueran suyos. A Javi, que llegaste justo en el momento que necesitaba ese empujón.

Al tutor de este proyecto, Víctor Fernández-Viagas Escudero, por su paciencia y dedicación durante todos estos meses. Siempre atento y dispuesto a ayudarme cuando fuera necesario.

Beatriz Pérez Barranco

Sevilla, 2020

Resumen

En este trabajo fin de grado (TFG) se aborda la adaptación de una serie de modelos matemáticos a un entorno productivo habitual. Se propone así un entorno de taller de flujo regular sin permutación, comparable a alguno de los entornos reales encontrados en la industria de manufactura. La adaptación se lleva a cabo mediante el desarrollo de los modelos de programación lineal entera mixta, llamados *MILP*, definiendo una función objetivo con una serie de restricciones para cada modelo. Este procedimiento se hace mediante programación en lenguaje C para automatizar los procesos y llamar al solver Gurobi para obtener los resultados. Por último, se estudiará el rendimiento de los modelos con respecto a diferentes parámetros.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE DIAGRAMAS

ÍNDICE DE GRÁFICAS

1 INTRODUCCIÓN

Este apartado trata de hacer una introducción al tema de la programación de la producción, así como poner en situación al lector sobre el objetivo de este proyecto y el contenido del documento.

Se introduce terminología en inglés, entre paréntesis y en cursiva, que se irá explicando conforme aparezca. Una vez se tienen los términos definidos, se utilizará frecuentemente la expresión en inglés o sus siglas por conveniencia. Se tomarán como referencia para estas definiciones (Framinan, Leisten and Ruiz García, 2014; Pérez, Fernández-Viagas and Framinan, 2017)

1.1 Programación de la producción

En términos generales, la programación y el control de la producción se inscriben dentro de la organización de la producción (*production management*)

La programación de la producción (*manufacturing scheduling*) es un proceso de toma de decisiones que se usa sobre todo en las industrias de fabricación y producción o en el sector servicios, así como en entornos de tratamiento de información. Se trata de la asignación de los trabajos o tareas a los recursos existentes o disponibles de la empresa. También es útil en el ámbito del transporte y de la distribución y en otro tipo de industrias de servicio.

Al aplicar esta asignación lo que se obtiene es un programa (*production schedule*) donde se identifica el trabajo que debe entrar en cada recurso y cuándo debe hacerlo.

Los recursos y los trabajos pueden definirse de diferente forma según el entorno. Sin embargo, se llega a un consenso para denominar "de forma genérica" dichos factores. Los productos fabricados o las unidades de trabajo en las que puede ser dividida una actividad de fabricación se denomina trabajo (*job*). A su vez, las operaciones o tareas requeridas por los trabajos están provistos por diferentes recursos productivos disponibles en el taller, y además se asume que es posible identificar dichas unidades de recursos como máquinas (*machines*)

Los recursos pueden ser máquinas en una fábrica, un operario o trabajador, una carretilla elevadora, pistas en un aeropuerto, etc. De igual forma, las tareas pueden ser operaciones en un proceso de producción, una plancha de metal, una caja, despegues y aterrizajes en el aeropuerto, etc.

El fin de un proceso de programación de la producción es encontrar, al menos, un programa admisible (*feasible schedule*) que minimice un cierto objetivo. Dicho programa debe cumplir con todas las restricciones y características del proceso. El método de obtención de un programa se denomina algoritmo. Una vez encontrado al menos un programa admisible, es el momento de elegir. Para elegirlo, se trata de definir uno o más objetivos o parámetros a alcanzar y así comparar los resultados obtenidos en cada programa. Los objetivos también pueden tomar diferentes formas según el ambiente de trabajo. A esta definición se le conoce como la función objetivo.

La aplicación de este proceso de programación de la producción en la "vida real" se lleva a cabo de diferente forma según la entidad que lo gestione. Dicha actividad incumbe a las diferentes áreas de una empresa (área de fabricación, de producción, compras, ventas, etc) y que deben tener una buena comunicación entre ellos para lograr sus objetivos. A menudo las empresas utilizan un sistema de información que es transversal a toda la empresa.

Una fábrica moderna, por ejemplo, suele tener un sistema de información complejo. Este se basa en un ordenador central y una base de datos. Las redes de los ordenadores y los terminales son utilizados para recibir información de la base de datos, así como introducir nueva información a esta. El software que controla este elaborado sistema de información se llama sistema ERP (*Enterprise Resource Planning*) y aunque muchas compañías desarrollan sus propios sistemas, el más conocido y utilizado actualmente es SAP. Estos sistemas ERP actúan como una red de información que atraviesa la empresa con enlaces a sistemas de apoyo para la toma de decisiones.

La programación de la producción se puede hacer de forma interactiva utilizando cualquier sistema de apoyo para la toma de decisiones que se instala en un terminal conectado al sistema ERP. Sin embargo, también hay entornos donde la comunicación entre la función de la programación y otras entidades de toma de decisión se hacen presencialmente en reuniones o mediante comunicados. En un futuro se prevé que la comunicación entre empresa y cliente o empresa y proveedor se haga enteramente mediante este tipo de sistemas. Todos los datos necesarios para hacer compras, ventas o gestiones estarán incluidos en el sistema.

Por último, en lo que se refiere a la representación gráfica, el método más utilizado para ilustrar un programa es el diagrama de Gantt. Este diagrama fue desarrollado por Henry Laurence Gantt en 1917. El diagrama proporciona una descripción gráfica y la programación temporal de todas las actividades y elementos de un proyecto. Se construye con un eje horizontal que representa la duración total del proyecto, dividido en incrementos temporales (días, semanas, meses…) En el eje vertical se pueden representar bien los recursos o bien los trabajos que componen el proyecto.

1.2 Objetivo del proyecto y estructura del documento

El objetivo de este proyecto es resolver el problema de forma óptima, con ayuda de los modelos de la literatura existente.

Para resolver el problema que se propone en este proyecto y encontrar el programa que logre el mejor resultado de la función objetivo, se llevarán a cabo una serie de adaptaciones de los modelos originales, descritos en el artículo (Stafford, Tseng and Gupta, 2005)

Para proceder con la resolución de los modelos, se lleva a cabo la programación en lenguaje C de cada uno de ellos. Con ayuda de esta programación en C también se automatizará la toma de datos de la batería de instancias dada y la generación de los archivos adecuados para su resolución con el software Gurobi. Por último, se hará un análisis de los resultados para concluir con el mejor modelo en cuanto a tiempo de cómputo y obtención de la mejor solución.

En cuanto a la estructura del proyecto se diferenciarán los siguientes puntos:

- En este primer apartado se hace una introducción al tema de la programación de operaciones. Se exponen los objetivos del proyecto y la estructura del documento.
- En el segundo apartado se describirá en profundidad el problema propuesto, además de mostrarse el problema original que se toma de base para el desarrollo del proyecto.
- El tercer apartado trata fundamentalmente la descripción de los modelos del problema original, así como de los modelos adaptados para que cumplan las restricciones del problema propuesto. Todos los modelos irán acompañados de un ejemplo explicativo para clarificar las diferencias entre ellos.
- En el cuarto, se lleva a cabo el proceso de resolución. En primer lugar, se explica la generación de los archivos necesarios para la ejecución con el software Gurobi. Se completa con un análisis de los resultados obtenidos, teniendo en cuenta varios parámetros como son la factibilidad de las soluciones, la obtención de la mejor solución o la desviación de las soluciones con respecto a la mejor encontrada.
- Por último, en el quinto apartado se expondrán las conclusiones obtenidas a partir del análisis de los resultados del apartado anterior.
- Además, se añade un anexo con el código en lenguaje de programación C de los modelos descritos.

2 CARACTERIZACIÓN DEL PROBLEMA

2.1 Términos generales

Para comenzar la caracterización, se definen una serie de términos esenciales para la comprensión del proyecto:

Las máquinas son recursos productivos que transforman un material. En este documento se hará referencia a este concepto con la letra M. Los trabajos son los productos que sufren transformación de las máquinas del taller. Se nombran con la letra N. Por último, cabe destacar el tiempo de proceso, variable que indica la duración del proceso de transformación de un trabajo dado en una máquina en concreto. Se denomina con T_{ij}.

Por otro lado, para trabajar con los modelos de programación de la producción (*scheduling model)*, conviene tener claras varias definiciones antes de entrar en la descripción del problema en concreto.

Un modelo es la abstracción de un problema real de decisión, puede ser de forma matemática, algorítmica, etc. Una vez se ha pasado del problema de programación de la producción al modelo, se lleva a cabo un método de resolución por el cual se obtiene la solución del modelo que se aplicará al caso inicial. Es importante recalcar la diferencia entre problema y modelo.

La solución que se obtiene para el problema de programación de la producción es un programa. De este programa se puede deducir la asignación de los trabajos a las máquinas y los instantes de inicio y fin de cada trabajo.

Para escribir un modelo de programación de operaciones se acude a la notación procedente de (Graham *et al.*, 1979) en tomando el esquema de clasificación de la *Figura 1*.

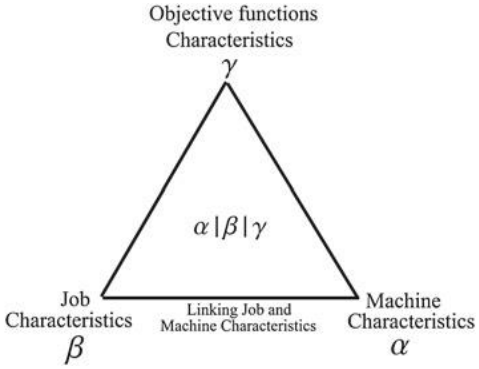


Figura 1. Clasificación α/β/γ de los modelos de programación de la producción

Se necesita la definición de α, β y γ para un modelo:

- Entorno (α): define las características de la máquina y puede estar formada por dos elementos: $\alpha = \alpha_1\alpha_2$
    - α_1 siendo un 1 si se refiere a un entorno de una sola máquina o una letra indicando la disposición de las máquinas en el caso de haber más de una. Por ejemplo, $\alpha_1: P$ se referiría a un entorno de máquinas idénticas en paralelo (*parallel machines*); $\alpha_1: F$ si se trata de un taller de flujo regular (*flowshop*); $\alpha_1: J$ cuando se tiene un entorno taller de trabajos (*job shop*).
    - α_2 siendo el número de máquinas en el caso de que sean un conjunto de máquinas en serie o en paralelo.

- Restricciones (β): indica las restricciones del proceso. En este caso se pueden tener restricciones o no. Si hay más de una restricción se separará con comas. En el caso de no haber restricciones hay unas suposiciones generales a considerar como son la disponibilidad de los trabajos al inicio del horizonte de la programación, la ininterrupción de los trabajos o la continua disponibilidad de las máquinas, así como que cada máquina puede hacer un solo trabajo al igual que un trabajo puede ser realizado solo en una máquina. Además, se supone el espacio de almacenamiento de productos (*buffer)* entre máquinas, infinito y el tiempo de transporte, despreciable.
- Criterio (γ): hace referencia a la función objetivo que se quiere alcanzar según el problema. Suelen estar enfocados al tiempo de finalización del proceso.

Antes de profundizar en la caracterización del problema, a continuación, se exponen dos ejemplos detallados de cómo pueden presentarse algunos modelos:

$$1 \mid r_j \mid \sum C_j$$

Este modelo representa un entorno en el que solo se dispone de una máquina, que tiene la restricción r_j que indica el instante a partir del que los trabajos están disponibles. Con respecto a la función objetivo que se define es $\sum C_j$ e indica la suma de los tiempos de finalización de cada trabajo.

$$F3 \mid prmu, no - idle \mid C_{max}$$

Para este ejemplo se tiene un entorno de tipo taller con 3 máquinas en serie donde todos los trabajos tienen que pasar por todas las máquinas. Las restricciones a cumplir son: todas las máquinas tienen la misma secuencia de trabajos y no puede haber tiempos ociosos de máquina entre los trabajos. La función objetivo definida es el mayor tiempo finalización de entre los trabajos procesados (*makespan*)

2.2 Caracterización

Como se ha expuesto anteriormente, para delimitar el problema objeto de estudio hay que definir primero las tres variables α|β|γ:

- Entorno (α: Fm): en este caso se tiene un entorno tipo taller de flujo regular (*Flowshop*) con M máquinas en serie. Cada trabajo debe pasar por todas las máquinas sucesivamente.
- Restricciones (β: ø): a parte de las suposiciones generales, no se va a tener en cuenta ninguna restricción adicional.
- Criterio (γ: C_{max}): la función objetivo más común es minimizar la duración total del programa (*makespan*) que se definirá de forma diferente según las variables que precise cada modelo.

En resumen, el problema propuesto queda definido tal que así: $Fm \mid\mid C_{max}$

Por otro lado, el problema que se toma de referencia se define se la siguiente forma: $Fm \mid prmu \mid C_{max}$

Como se puede observar, la diferencia fundamental es la restricción de permutación existente en el modelo de apoyo. Para explicar la diferencia es importante tener claras varias definiciones sobre dichos entornos:

- En los entornos de tipo taller de flujo regular (*Flowshop*) las máquinas se disponen en serie y todos los trabajos tienen que pasar por todas las máquinas con una ruta predeterminada. La ruta es el orden de máquinas que tiene que seguir un trabajo para ser procesado.
- Sin embargo, la secuencia de todas las máquinas puede ser la misma (*Flowshop de permutación*) o diferente (*Flowshop sin permutación*) La secuencia es el orden en que los trabajos son procesados en una máquina.

El hecho de plantear este caso: $Fm \mid\mid C_{max}$, hace que el problema se acerque lo máximo posible a cualquier caso real. No en todos los entornos industriales se tienen iguales secuencias en todas las máquinas.

3 DESCRIPCIÓN DE LOS MODELOS

3.1 Introducción

Durante este apartado se expone la notación utilizada junto a la definición de un ejemplo sencillo que se utilizará para esclarecer cada modelo utilizado. Además, se describen los modelos de programación lineal entera que se proponen para resolver el problema original, así como los modelos adaptados al problema propuesto. Antes de adaptar los modelos al problema en cuestión, se ha trabajado en profundidad el problema original, definido como: $Fm \mid prmu \mid C_{max}$, de acuerdo con lo descrito en (Stafford, Tseng and Gupta, 2005). Otra literatura de referencia utilizada para el desarrollo ha sido (Pinedo, 2012) y (Framinan, Leisten and Ruiz García, 2014)

En los modelos de programación de la producción se puede observar la complejidad que van adquiriendo los problemas según se vayan aumentando el número de trabajos, N, y el número de máquinas, M. Además, al añadir restricciones se aumenta la complejidad del problema y pueden aparecer soluciones no factibles.

En general, para el entorno *Flowshop*, el número de soluciones está acotada: si la restricción es *Flowshop de permutación*, el número de soluciones será $N!$ y si la restricción es de *Flowshop sin permutación*, el número de soluciones se incrementa notablemente: $(N!)^M$

Debido al intratable número de soluciones se acude a la teoría de complejidad computacional que se desarrolló en los años 70. Consiste en la clasificación de los problemas computacionales de acuerdo a su estructura y dificultad. En cuanto a la clasificación de los modelos, estos pueden ser modelos con algoritmos polinomiales (*P*) o no polinomiales (*NP-hard*)

Se utilizan los modelos de programación lineal entera mixta (*mixed-integer linear programing (MILP)*) para solucionar de forma eficiente un problema de este tipo. De esta forma, estos problemas se consideran como un híbrido entre varias categorías de modelamiento, siendo el caso más típico el de mezclar variables enteras, binarias y variables continuas. El tiempo computacional es prácticamente despreciable en comparación con lo que se puede tardar en solucionar un solo problema manualmente.

Relacionado con la complejidad de los problemas y ahondando en los modelos matemáticos de las familias de Wagner y Manne, en el análisis de los modelos de (Stafford, Tseng and Gupta, 2005) se explica que la complejidad se usa para referirse a lo "grande" que se convierte cada problema, en términos de número de variables reales requeridas, variables binarias enteras y restricciones como una función de M y N. Dependiendo de cada uno de los aspectos, cada modelo o cada familia de modelos tiene unas características diferentes y son más o menos complejos.

3.2 Notación matemática

La notación que se utilizará sigue la notación convencional extraída de (Stafford, Tseng and Gupta, 2005)

Los datos considerados son:

- M, siendo el número de máquinas.
- N, siendo los trabajos.
- T_{ri}, indica el tiempo de proceso del trabajo i en la máquina r.
- P, representando un valor suficientemente grande como para asegurar que se cumple solamente una relación del par de restricciones dicotómicas. Para calcularlo: $P = 100 \times N$

Los índices usados son:

- r, para las máquinas $(1 \leq r \leq M)$
- i, k, para los trabajos $(1 \leq i, k \leq N)$
- j, para la posición en la secuencia $(1 \leq j \leq N)$

Las variables serán las siguientes:

- B_{rj}, tiempo de comienzo del trabajo de la posición j en la máquina r
- C_{ri}, tiempo de finalización del trabajo i en la máquina r
- $D_{ik} = \begin{cases} 1\ si\ el\ trabajo\ i\ se\ programa\ en\ cualquier\ momento\ antes\ del\ trabajo\ k \\ 0\ eoc \end{cases}$
- E_{rj}, tiempo final del trabajo de la posición j en la máquina r
- S_{ri}, tiempo de inicio del trabajo i en la máquina r
- X_{rj}, tiempo ocioso de la máquina r antes del inicio del trabajo en la posición j de la secuencia
- Y_{rj}, tiempo de espera del trabajo en la posición j de la secuencia tras ser procesado completamente en la máquina r
- $Z_{ij} = \begin{cases} 1\ si\ el\ trabajo\ i\ se\ asigna\ a\ la\ posicion\ j\ de\ la\ secuencia \\ 0\ eoc \end{cases}$

Sin embargo, al hacer las adaptaciones de los modelos al problema propuesto, se necesitan añadir, en las variables de base, nuevos subíndices:

- $D_{ikr} = \begin{cases} 1\ si\ el\ trabajo\ i\ se\ programa\ en\ cualquier\ momento\ antes\ del\ trabajo\ k, \\ para\ la\ máquina\ r \\ 0\ eoc \end{cases}$
- $Z_{ijr} = \begin{cases} 1\ si\ el\ trabajo\ i\ se\ asigna\ a\ la\ posicion\ j\ de\ la\ secuencia, en\ la\ máquina\ r \\ 0\ eoc \end{cases}$

3.3 Ejemplo ilustrativo

Para ilustrar el proceso de adaptación de los modelos se va a utilizar una instancia sencilla como ejemplo. Esta instancia será resuelta según los modelos originales para el problema de base, y también según los modelos adaptados para el problema propuesto. Aunque se parta de los mismos datos y se obtenga la misma función objetivo, la secuencia obtenida no tiene por qué coincidir, al igual que los valores de las variables utilizadas en los modelos pueden ser diferentes.

Se ha tomado una instancia de ejemplo con los siguientes parámetros:

- $M = 3$
- $N = 7$
- Tiempos de proceso (T_{ri}) generados aleatoriamente entre 1 y 10 unidades temporales. Dichos tiempos de proceso se encuentran recogidos en la *Tabla 1*.

	J1	**J2**	**J3**	**J4**	**J5**
M1	2	6	7	6	10
M2	5	1	10	8	6
M3	10	1	4	2	1

Tabla 1. Tiempos de proceso de la instancia de ejemplo

3.4 Modelos originales de programación lineal entera

Son cinco los modelos matemáticos originales en los que se basa este proyecto. Dichos modelos se agrupan en dos familias: tres modelos forman la familia de modelos de Wagner: WST, Wilson y TS2 y los dos modelos restantes, la familia Manne: SGST y LYeq.

En el caso de la familia de modelos de Wagner, el problema es el de asignación de los trabajos en las posiciones. Sin embargo, en la familia de Manne se plantea el problema de la asignación de los trabajos en posiciones de la secuencia de producción, para lo que utiliza restricciones dicotómicas.

A continuación, se definirán cada uno de los modelos acompañados de la resolución de su ejemplo para mayor claridad.

3.4.1 Familia de modelos de Wagner

Para empezar, esta familia tiene dos restricciones comunes además de las especificadas según modelo.

$$\sum_{j=1}^{N} Z_{ij} = 1 \qquad (1 \leq i \leq N) \tag{1}$$

$$\sum_{i=1}^{N} Z_{ij} = 1 \qquad (1 \leq j \leq N) \tag{2}$$

Las restricciones (1) y (2) aseguran, respectivamente, que cada trabajo está asignado a una sola posición en la secuencia y que cada posición tiene asignada únicamente un trabajo.

3.4.1.1 Modelo WST

El modelo WST propone minimizar la función objetivo, C_{max}, añadiendo una serie de restricciones utilizando las variables X_{rj}, Y_{rj} y Z_{ij}.

$$Min\ C_{max} = \sum_{i=1}^{N} T_{Mi} + \sum_{p=1}^{N} X_{Mp} \tag{1.1}$$

$s.a$:

$$\sum_{i=1}^{N} T_{ri} Z_{i,j+1} + X_{r,j+1} + Y_{r,j+1} = \sum_{i=1}^{N} T_{r+1,i} Z_{ij} + X_{r+1,j+1} + Y_{rj}$$
$$(1 \leq r \leq M-1; 1 \leq j \leq N-1) \tag{1.2}$$

$$X_{r+1,1} = X_{r,1} + Y_{r,1} + \sum_{i=1}^{N} T_{ri} Z_{i1} \quad (1 \leq r \leq M-1) \tag{1.3}$$

$$Y_{r1} = 0 \quad (1 \leq r \leq M-1) \tag{1.4}$$

$$X_{rj}, Y_{rj} \geq 0 \qquad (1 \leq r \leq M, 1 \leq j \leq N)$$

$$Z_{ij} \in \{0,1\} \qquad (1 \leq i, j \leq N)$$

La función objetivo se rige según la ecuación (1.1) y se define como la suma de tiempos de proceso de todos los trabajos en la última máquina M más la suma de tiempos ociosos de dicha máquina.

De las restricciones dadas se asegura lo siguiente: de acuerdo con las ecuaciones (1.2) y (1.3) el trabajo en la posición de la secuencia $j+1$ no se puede comenzar a procesar en la maquina r hasta que el trabajo de la posición j se haya completado en la misma máquina, así mismo, el trabajo en la posición j no puede empezar a procesarse en la máquina $r+1$ hasta que haya terminado de procesarse en la maquina r.

Por último, con la restricción (1.4) se fuerza al trabajo de la posición 1 a ser procesado inmediatamente después en las máquinas siguientes, de la 2 a la M una vez se haya ido completando su proceso en las máquinas previas.

Resolución del ejemplo:

Al solucionar el ejemplo mediante el modelo original WST se obtienen los resultados recogidos en la *Tabla 2* de las variables que caracterizan al modelo.

En este caso, por ejemplo, se obtiene $X_{23} = 0$, lo que significa que el trabajo de la posición 3 no debe esperar para comenzar a procesarse en la máquina 2. Sin embargo, en el caso de $X_{32} = 7$ se obtiene un valor diferente de cero, lo que quiere decir que el trabajo de la posición 2 debe esperar 7 unidades de tiempo para poder procesarse en la máquina 3.

En cuanto a la variable Y, también puede tomar valores iguales a cero o no: $Y_{14} = 1$ significa que el trabajo de la posición 4 debe esperar 1 unidad de tiempo tras procesarse en la máquina 1 antes de procesarse en la máquina siguiente.

Por último, los valores de Z solo pueden ser 1 o 0, en este caso solo se ha expuesto aquellas variables que son iguales a 1, puesto que con ellas se saca la secuencia de los trabajos en las máquinas: $Z_{25} = 1$ indica que el trabajo 2 se asigna a la posición 5. En este caso se obtiene la secuencia: 1, 4, 3, 5, 2.

$X_{11} = 0$	$X_{12} = 0$	$X_{13} = 0$	$X_{14} = 0$	$X_{15} = 0$	$Y_{11} = 0$	$Y_{12} = 0$	$Y_{13} = 0$	$Y_{14} = 1$	$Y_{15} = 0$
$X_{21} = 2$	$X_{22} = 1$	$X_{23} = 0$	$X_{24} = 0$	$X_{25} = 0$	$Y_{21} = 0$	$Y_{22} = 8$	$Y_{23} = 0$	$Y_{24} = 0$	$Y_{25} = 0$
$X_{31} = 7$	$X_{32} = 7$	$X_{33} = 0$	$X_{34} = 2$	$X_{35} = 0$	$Z_{11} = 1$	$Z_{25} = 1$	$Z_{33} = 1$	$Z_{42} = 1$	$Z_{54} = 1$

Tabla 2. Valores obtenidos para las variables del modelo WST original

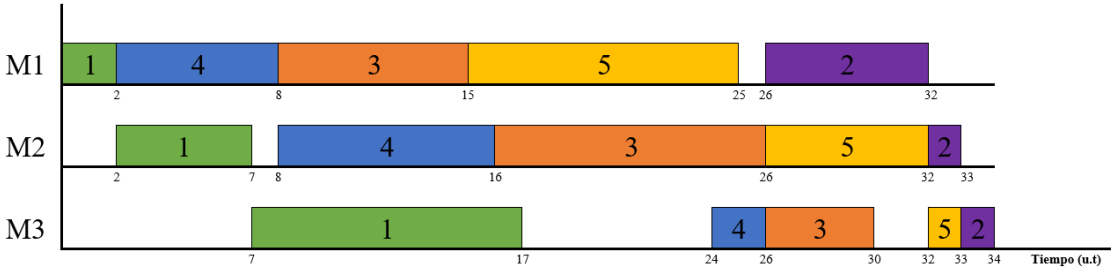


Diagrama 1. Gantt de la solución según el modelo WST original

A partir de los valores obtenidos tras solucionar el problema con el modelo original de WST, se crea el diagrama de Gantt de forma que se tiene la solución visual recogida en el *Diagrama 1*.

3.4.1.2 Modelo Wilson

El modelo Wilson propone minimizar la función objetivo, *makespan*, definido como el tiempo de inicio del último trabajo en la última máquina más su tiempo de proceso según (2.1)

$$Min\ B_{MN} + \sum_{i=1}^{N} T_{Mi} Z_{iN} \tag{2.1}$$

$s.a$:

$$B_{1j} + \sum_{i=1}^{N} T_{1i} Z_{ij} \leq B_{1,j+1} \quad (1 \leq j \leq N-1) \tag{2.2}$$

$$B_{11} = 0 \tag{2.3}$$

$$B_{r1} + \sum_{i=1}^{N} T_{ri} Z_{i1} \leq B_{r+1,1} \quad (1 \leq r \leq M-1) \tag{2.4}$$

$$B_{rj} + \sum_{i=1}^{N} T_{ri} Z_{ij} \leq B_{r+1,j} \quad (1 \leq r \leq M-1; 2 \leq j \leq N) \tag{2.5}$$

$$B_{rj} + \sum_{i=1}^{N} T_{ri} Z_{ij} \leq B_{r,j+1} \quad (2 \leq r \leq M; 1 \leq j \leq N-1) \tag{2.6}$$

$$B_{rj} \geq 0 \qquad (1 \leq r \leq M, 1 \leq j \leq N)$$

$$Z_{ij} \in \{0,1\} \qquad (1 \leq i, j \leq N)$$

Con la restricción (2.2) se fuerza a todos los trabajos a comenzar en la máquina 1 justo después de que sus trabajos predecesores hayan sido procesados completamente en la maquina dada. La restricción (2.3) fuerza a que el primer trabajo de la secuencia empiece su proceso en la máquina 1 en el instante inicial. De la ecuación (2.4) se deduce el forzado del primer trabajo en la secuencia a ser inmediatamente procesado en la siguiente máquina una vez se haya procesado en la actual.

La ecuación (2.5) asegura que, para todas las máquinas, cada uno de los trabajos no puede empezar en la próxima máquina hasta que se haya completado en la máquina actual. Por último, la restricción (2.6) asegura que, para todas las máquinas, un trabajo no puede empezar a ser procesado en una máquina hasta que el trabajo anterior en la secuencia haya sido completamente procesado en dicha máquina.

Resolución del ejemplo:

En esta ocasión, el ejemplo se resuelve mediante el modelo original de WILSON. Los resultados recogidos en la *Tabla 3* de las variables que caracterizan al modelo son los obtenidos tras la aplicación de dicho modelo.

Ahora, la variable que caracteriza el modelo sería B_{rj} pudiendo tomar cualquier valor: $B_{13} = 8$ indica que, para el trabajo en la posición 3 en la máquina 1, su tiempo de inicio es 8 unidades de tiempo.

Al igual que antes, los valores de Z solo pueden ser 1 o 0: $Z_{33} = 1$ indica que el trabajo 3 se asigna a la posición 3. De la misma forma se obtiene la secuencia: 1, 4, 3, 5, 2.

$B_{11} = 0$	$B_{12} = 2$	$B_{13} = 8$	$B_{14} = 15$	$B_{15} = 25$	$Z_{11} = 1$	$Z_{25} = 1$	$Z_{33} = 1$	$Z_{42} = 1$	$Z_{54} = 1$
$B_{21} = 2$	$B_{22} = 8$	$B_{23} = 16$	$B_{24} = 26$	$B_{25} = 32$					
$B_{31} = 7$	$B_{32} = 26$	$B_{33} = 28$	$B_{34} = 32$	$B_{35} = 33$					

Tabla 3. Valores obtenidos para las variables del modelo WILSON original

A partir de los valores obtenidos tras solucionar el problema con el modelo original de WILSON, se crea el

diagrama de Gantt de forma que se tiene la solución visual recogida en la *Diagrama 2*.

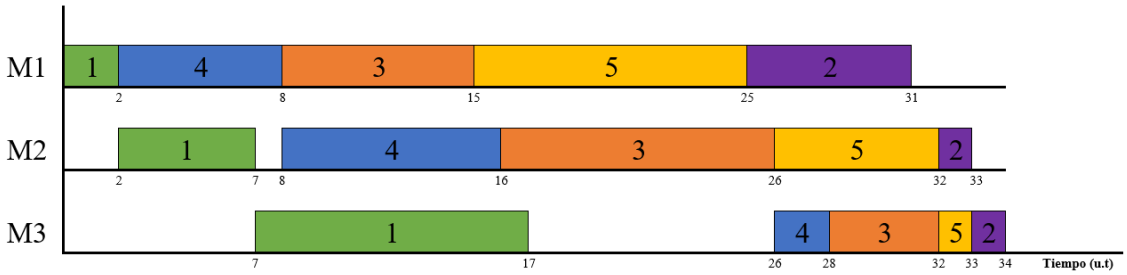


Diagrama 2. Gantt de la solución según el modelo WILSON original

3.4.1.3 Modelo TS2

El modelo TS2 propone minimizar la función objetivo, que en este caso se toma como el tiempo de finalización del último trabajo en la última máquina (3.1)

$$Min\ E_{MN} \tag{3.1}$$

$s.a$:

$$E_{rj} + \sum_{i=1}^{N} T_{ri} Z_{i,j+1} \leq E_{r,j+1} \quad (1 \leq r \leq M; 1 \leq j \leq N-1) \tag{3.2}$$

$$E_{rj} + \sum_{i=1}^{N} T_{r+1,i} Z_{ij} \leq E_{r+1,j} \quad (1 \leq r \leq M-1; 1 \leq j \leq N) \tag{3.3}$$

$$E_{11} \geq \sum_{i=1}^{N} T_{1i} Z_{i1} \tag{3.4}$$

$$E_{rj} \geq 0 \qquad (1 \leq r \leq M, 1 \leq j \leq N)$$

$$Z_{ij} \in \{0,1\} \qquad (1 \leq i,j \leq N)$$

La restricción (3.2) asegura que el trabajo de la posición $j+1$ en la secuencia no puede finalizar en ninguna máquina hasta que el trabajo de la posición anterior de la secuencia haya sido completado en esa máquina y el trabajo de la posición $j+1$ haya sido también procesado en esa máquina.

La ecuación (3.3) asegura que un trabajo dado no puede terminar en la máquina $r+1$ hasta que al menos no termine en la maquina anterior y entonces sea totalmente procesado en la máquina $r+1$.

Por último, la ecuación (3.4) declara que el trabajo de la posición 1 en la secuencia no puede terminar en la máquina 1 hasta que al menos haya sido procesado en dicha máquina.

Resolución del ejemplo:

En el caso de la resolución del ejemplo mediante el modelo TS2, los resultados recogidos en la Tabla 4 son los de las variables E_{rj} que caracterizan el modelo, pudiendo tomar cualquier valor: $E_{23} = 26$ representa, para el trabajo en la posición 3 en la máquina 2, su tiempo de finalización de proceso, que es 26 unidades de tiempo.

De la misma forma, los valores de Z solo pueden ser 1 o 0: $Z_{42} = 1$ indica que el trabajo 4 se asigna a la posición 2. De nuevo se repite la secuencia obtenida anteriormente: 1, 4, 3, 5, 2.

$E_{11} = 2$	$E_{12} = 8$	$E_{13} = 15$	$E_{14} = 25$	$E_{15} = 32$	$Z_{11} = 1$	$Z_{25} = 1$	$Z_{33} = 1$	$Z_{42} = 1$	$Z_{54} = 1$
$E_{21} = 7$	$E_{22} = 16$	$E_{23} = 26$	$E_{24} = 32$	$E_{25} = 33$					
$E_{31} = 18$	$E_{32} = 20$	$E_{33} = 32$	$E_{34} = 33$	$E_{35} = 34$					

Tabla 4. Valores obtenidos para las variables del modelo TS2 original

A partir de los valores obtenidos tras solucionar el problema con el modelo original de TS2, se crea el diagrama de Gantt de forma que se tiene la solución recogida en el *Diagrama 3*.

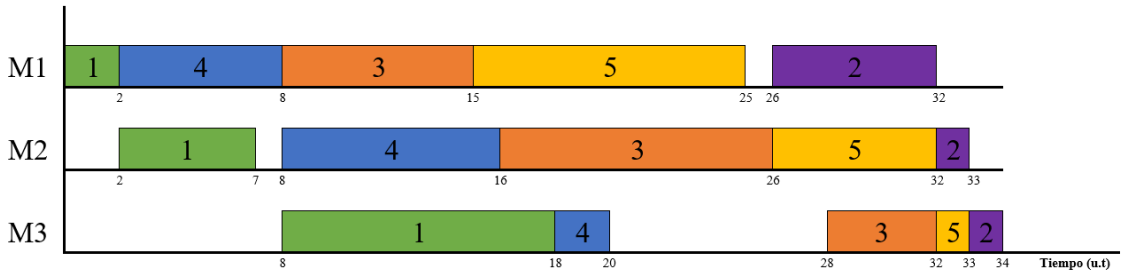


Diagrama 3. Gantt de la solución según el modelo TS2 original

3.4.2 Familia de modelos de Manne

En este caso se utilizan los modelos pertenecientes a esta familia, SGST y LYeq. La familia de modelos Manne utiliza pares de restricciones dicotómicas para controlar el orden relativo de los trabajos en la secuencia de producción. Los modelos SGST utilizan las restricciones dicotómicas originales de Manne y los modelos LY utilizan la modificación de Liao-You.

Además, en comparación con la familia de Wagner, en este caso se utilizarán más restricciones para problemas del mismo tamaño y, por tanto, los modelos de Wagner son capaces de resolver problemas más complejos en un menor tiempo computacional.

En la familia de modelos de Manne, no se usa el término "posición del trabajo" por lo que, básicamente se trabaja solo con trabajos y máquinas, utilizando los índices i y k para relacionar los trabajos entre ellos y el índice r para indicar la máquina en cuestión.

3.4.2.1 Modelo SGST

Para el modelo SGST se define la función objetivo, siendo el *makespan*, directamente como C_{max}:

$$Min\ C_{MAX} \tag{4.1}$$

$s.a$:

$$C_{1i} \geq T_{1i} \quad (1 \leq i \leq N) \tag{4.2}$$

$$C_{r+1,i} - C_{ri} \geq T_{r+1,i} \quad (1 \leq r \leq M-1; 1 \leq i \leq N) \tag{4.3}$$

$$C_{ri} - C_{rk} + PD_{ik} \geq T_{ri} \quad (1 \leq r \leq M; 1 \leq i < k \leq N) \tag{4.4}$$

$$C_{rk} - C_{ri} + P(1 - D_{ik}) \geq T_{rk} \quad (1 \leq r \leq M; 1 \leq i < k \leq N) \tag{4.5}$$

$$C_{MAX} \geq C_{Mi} \quad (1 \leq i \leq N) \tag{4.6}$$

$$C_{ri}, C_{MAX} \geq 0 \qquad (1 \leq r \leq M, 1 \leq i \leq N)$$

$$D_{ik} \in \{0,1\} \qquad (1 \leq i < k \leq N)$$

En cuanto a las restricciones, (4.2) indica que ningún trabajo pueda terminar en la primera máquina antes de ser completamente procesado en ella. La ecuación (4.3) expresa que cada trabajo no puede ser completado en la máquina $r+1$ hasta que no haya sido completado en la máquina anterior y completamente procesado en la máquina actual, en conclusión, esto quiere decir que cada trabajo solo puede ser procesado en una máquina al mismo tiempo.

El siguiente par de ecuaciones, (4.4) y (4.5) forman los pares de restricciones dicotómicas que relacionan cada par de trabajos i y k ($i \neq k$), de modo que cualquiera de los trabajos i o precede al k, o sigue al k en la secuencia, pero no ambas cosas.

Por último, la restricción (4.6) trata de asegurar que el *makespan* del programa es, al menos, igual que el tiempo de finalización del último trabajo de la secuencia.

Resolución del ejemplo:

Para el modelo SGST, los resultados son los recogidos en la *Tabla 5*. En este caso se obtienen los valores de las variables C_{ri} que indican el tiempo de finalización del trabajo i en la máquina r.

Por ejemplo, el valor de la variable $C_{24} = 16$ significa que el trabajo 4 en la máquina 2 termina de procesarse en 16 unidades de tiempo.

Además, se saca la secuencia gracias a las variables D_{ik}, que toman valores de 0 o 1 según si se cumple que el trabajo i se sitúe antes del trabajo k o no. En el caso de $D_{13} = 1$ significa que el trabajo 1 se encuentra antes del 3 en la secuencia. El valor de $D_{45} = 0$ quiere decir que es el trabajo 5 el que está por delante del trabajo 4. De esta forma se obtiene la secuencia: 1, 4, 3, 5, 2.

$C_{11} = 2$	$C_{12} = 31$	$C_{13} = 15$	$C_{14} = 8$	$C_{15} = 25$	$D_{12} = 1$	$D_{13} = 1$	$D_{14} = 1$	$D_{15} = 1$	$D_{35} = 1$
$C_{21} = 7$	$C_{22} = 33$	$C_{23} = 26$	$C_{24} = 16$	$C_{25} = 32$	$D_{45} = 1$	$D_{23} = 0$	$D_{24} = 0$	$D_{25} = 0$	$D_{34} = 0$
$C_{31} = 17$	$C_{32} = 34$	$C_{33} = 32$	$C_{34} = 19$	$C_{35} = 33$					

Tabla 5. Valores obtenidos para las variables del modelo SGST original

A partir de los valores obtenidos tras solucionar el problema con el modelo original de SGST, se crea el diagrama de Gantt de forma que se tiene la solución visual recogida en el *Diagrama 4*.

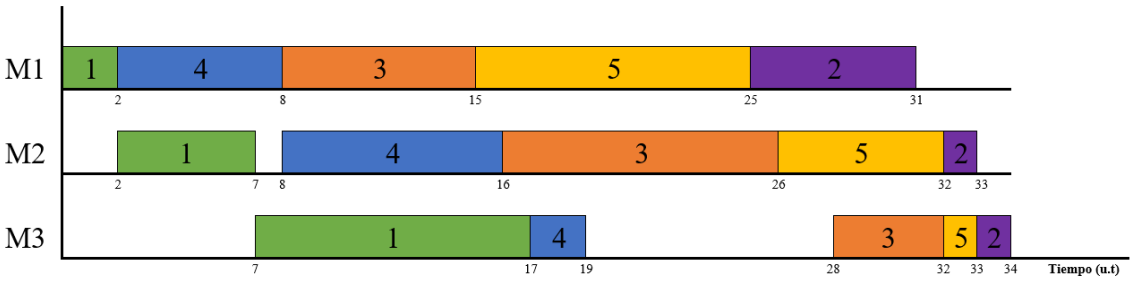


Diagrama 4. Gantt de la solución según el modelo SGST original

3.4.2.2 Modelo LYeq

Este modelo tiene en común con el anterior las restricciones siguientes: (5.2), (5.3), (5.4) además de la función objetivo (5.1). Y las dos ecuaciones restantes (5.5) y (5.6) se refieren de la misma forma, al par de restricciones dicotómicas.

$$Min\ C_{MAX} \tag{5.1}$$

$s.a$:

$$C_{1i} \geq T_{1i} \quad (1 \leq i \leq N) \tag{5.2}$$

$$C_{r+1,i} - C_{ri} \geq T_{r+1,i} \quad (1 \leq r \leq M-1; 1 \leq i \leq N) \tag{5.3}$$

$$C_{MAX} \geq C_{Mi} \quad (1 \leq i \leq N) \tag{5.4}$$

$$PD_{ik} + C_{ri} - C_{rk} - T_{ri} = Q_{rik} \quad (1 \leq r \leq M; 1 \leq i < k \leq N) \tag{5.5}$$

$$Q_{rik} \leq P - T_{ri} - T_{rk} \quad (1 \leq r \leq M; 1 \leq i < k \leq N) \tag{5.6}$$

$$C_{ri}, C_{MAX}, Q_{rik} \geq 0 \qquad (1 \leq r \leq M, 1 \leq i < k \leq N)$$

$$D_{ik} \in \{0,1\} \qquad (1 \leq i < k \leq N)$$

Resolución del ejemplo:

Ahora se resuelve según el modelo LYeq original que cuenta con las mismas variables que el modelo SGST para definirse. Los resultados se muestran en la *Tabla 6*.

Para las variables C_{ri} se obtienen valores como los siguientes: C_{24}=16 significa que el trabajo 4 en la máquina 2 termina en 16 unidades de tiempo.

Además, las variables D_{ik}, toman valores de 0 o 1 según si se cumple que el trabajo i se sitúe antes del trabajo k o no. Al igual que en el caso de la resolución según el modelo SGST, gracias a esta variable se obtiene la secuencia para todas las máquinas. En el caso de $D_{13} = 1$ significa que el trabajo 1 se encuentra antes del 3 en la secuencia. El valor de $D_{45} = 0$ quiere decir que es el trabajo 5 el que está por delante del trabajo 4. De esta forma se obtiene la secuencia: 1, 4, 3, 5, 2.

$C_{11} = 2$	$C_{12} = 32$	$C_{13} = 15$	$C_{14} = 8$	$C_{15} = 25$	$D_{12} = 1$	$D_{13} = 1$	$D_{14} = 1$	$D_{15} = 1$	$D_{35} = 1$
$C_{21} = 8$	$C_{22} = 33$	$C_{23} = 26$	$C_{24} = 16$	$C_{25} = 32$	$D_{45} = 1$	$D_{23} = 0$	$D_{24} = 0$	$D_{25} = 0$	$D_{34} = 0$
$C_{31} = 24$	$C_{32} = 34$	$C_{33} = 30$	$C_{34} = 26$	$C_{35} = 33$					

Tabla 6. Valores obtenidos para las variables del modelo LYeq original

A partir de los valores obtenidos tras solucionar el problema con el modelo original de LYeq, se crea el diagrama de Gantt asociado. Es representado en *Diagrama 5*.

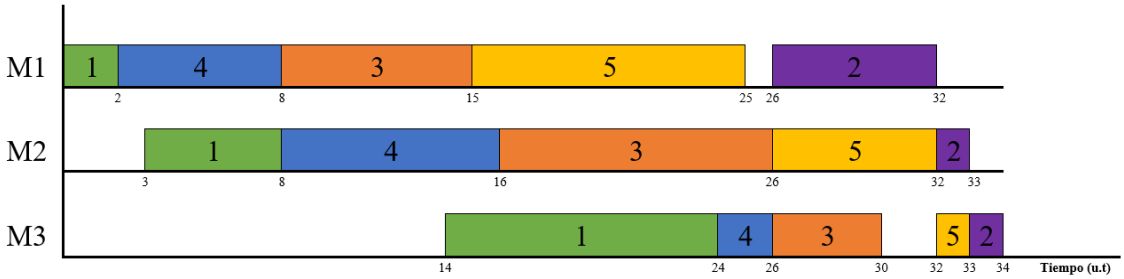


Diagrama 5. Gantt de la solución según el modelo LYeq original

3.5 Modelos adaptados al problema

A partir de los modelos originales se ha llegado a la adaptación de cuatro de ellos. En la familia de modelos de Wagner se encontrarán los modelos de WILSON y TS2 y, por lo que respecta a la familia de modelos de Manne se tienen los modelos SGST y LYeq.

La adaptación de estos modelos se ha llevado a cabo de forma que se alteraran lo menos posible y no se volvieran más complejos. Se ha tratado de no añadir restricciones ni variables diferentes a las existentes en los modelos originales. Este es el motivo principal por el que en este apartado no se cuenta con el primer modelo de Wagner descrito anteriormente, el modelo WST.

3.5.1 Familia de modelos de Wagner

Para adaptar los modelos originales, que están particularizados para el caso de *Flowshop de permutación*, hay que añadir la condición de que cada máquina tiene una secuencia diferente. Las restricciones comunes, quedan de la siguiente forma:

$$\sum_{j=1}^{N} Z_{ijr} = 1 \qquad (1 \leq i \leq N; 1 \leq r \leq M) \tag{1}$$

$$\sum_{i=1}^{N} Z_{ijr} = 1 \qquad (1 \leq j \leq N; 1 \leq r \leq M) \tag{2}$$

Ahora, a las restricciones originales de los modelos de Wagner basadas en que cada trabajo está asignado a una sola posición en la secuencia, y, que cada posición tiene asignada únicamente un trabajo, se le añade que dicha condición se debe cumplir para cada una de las máquinas. Esto se traduce en que, para cada máquina hay una matriz Z_{ij} diferente. El subíndice r, utilizado anteriormente, es el indicador para las máquinas.

3.5.1.1 Modelo Wilson

Para poder adaptar el modelo de Wilson, se ha modificado la ecuación (2.5) del modelo original añadiendo un término en la desigualdad que hace que el paréntesis se anule cuando los subíndices j_1 y j_2 coincidan, es decir, cuando el trabajo i ocupe la misma posición en ambas máquinas r y $r+1$, siendo P la variable anteriormente definida como un valor lo suficientemente grande como para asegurar que se cumple solamente una relación del par de restricciones dicotómicas.

$$Min\ B_{MN} + \sum_{i=1}^{N} T_{Mi} Z_{iNM} \tag{2.1}$$

$s.a$:

$$B_{1j} + \sum_{i=1}^{N} T_{1i} Z_{ij1} = B_{1,j+1} \quad (1 \le j \le N-1) \tag{2.2}$$

$$B_{11} = 0 \tag{2.3}$$

$$B_{r1} + \sum_{i=1}^{N} T_{ri} Z_{i1r} = B_{r+1,1} \quad (1 \le r \le M-1) \tag{2.4}$$

$$B_{r,j1} + T_{r,i} Z_{i,j1,r} \le B_{r+1,j2} + P\left(2 - Z_{i,j1,r} - Z_{i,j2,r+1}\right) \quad (1 \le i, r \le M-1, 1 \le j1, j2 \le N) \tag{2.5'}$$

$$B_{rj} + \sum_{i=1}^{N} T_{ri} Z_{ijr} \le B_{r,j+1} \quad (2 \le r \le M; 1 \le j \le N-1) \tag{2.6}$$

$$B_{rj} \ge 0 \qquad (1 \le r \le M, 1 \le j \le N)$$

$$Z_{ijr} \in \{0,1\} \qquad (1 \le r \le M, 1 \le i, j \le N)$$

Resolución del ejemplo:

Tras la modificación del modelo WILSON enfocado a la resolución del problema propuesto, se aplica para obtener numéricamente la solución óptima del ejemplo utilizado anteriormente. Los resultados están recogidos en la *Tabla 7*.

En cuanto a las variables que se utilizan para caracterizar la solución son las siguientes: la variable B_{rj} igual que en el modelo original y la variable Z_{ijr} que tomará valor 1 cuando cumpla que el trabajo i va en la posición j en la máquina r, por tanto, a diferencia del modelo original, con ayuda de esa variable se definen las secuencias para cada una de las máquinas.

En el caso de este problema ejemplo se obtienen las siguientes secuencias: para las máquinas 1 y 2, los trabajos se procesan en el orden 1,4,3,5,2; y para la máquina 3 se obtiene la secuencia 1,3,4,5,2.

$B_{11} = 0$	$B_{12} = 2$	$B_{13} = 8$	$B_{14} = 15$	$B_{15} = 25$	$Z_{111}=1$	$Z_{251}=1$	$Z_{331}=1$	$Z_{421}=1$	$Z_{541}=1$
$B_{21} = 2$	$B_{22} = 8$	$B_{23} = 16$	$B_{24} = 26$	$B_{25} = 32$	$Z_{112}=1$	$Z_{252}=1$	$Z_{332}=1$	$Z_{422}=1$	$Z_{542}=1$
$B_{31} = 7$	$B_{32} = 26$	$B_{33} = 30$	$B_{34} = 32$	$B_{35} = 33$	$Z_{113}=1$	$Z_{253}=1$	$Z_{323}=1$	$Z_{433}=1$	$Z_{543}=1$

Tabla 7. Valores obtenidos para las variables del modelo WILSON modificado

A continuación, se expone el diagrama de Gantt representado partir de los valores obtenidos tras solucionar el

problema con el modelo adaptado de WILSON.

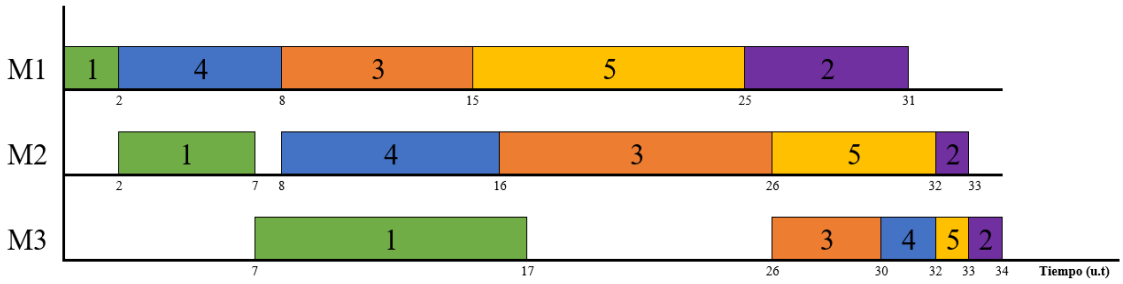


Diagrama 6. Gantt de la solución según el modelo WILSON modificado

3.5.1.2 Modelo TS2

En el caso del modelo TS2 se ha utilizado la misma técnica que para el modelo Wilson, se modifica la restricción (3.3) añadiendo el término que se cancela cuando coincide que el trabajo i se encuentra en la misma posición para máquinas sucesivas.

$$Min\ E_{MN} \tag{3.1}$$

$s.a$:

$$E_{rj} + \sum_{i=1}^{N} T_{ri} Z_{i,j+1,r} \leq E_{r,j+1} \quad (1 \leq r \leq M; 1 \leq j \leq N-1) \tag{3.2}$$

$$E_{r,j1} + T_{r+1,i} Z_{i,j2,r+1} \leq E_{r+1,j2} + P\left(2 - Z_{i,j1,r} - Z_{i,j2,r+1}\right) \quad (1 \leq i,r \leq M-1, 1 \leq j1, j2 \leq N) \tag{3.3'}$$

$$E_{11} \geq \sum_{i=1}^{N} T_{1i} Z_{i11} \tag{3.4}$$

$$E_{rj} \geq 0 \quad (1 \leq r \leq M, 1 \leq j \leq N)$$

$$Z_{ijr} \in \{0,1\} \quad (1 \leq r \leq M, 1 \leq i,j \leq N)$$

Resolución del ejemplo:

Al solucionar el ejemplo mediante el modelo TS2 modificado para el caso de *Flowshop sin permutación*, los resultados obtenidos son los recogidos en la Tabla 8.

Las variables son: E_{rj}, igual que el modelo original y Z_{ijr} que difiere del modelo original pues se le añade un subíndice que indica que para cada máquina se puede obtener una secuencia diferente.

Las secuencias de las máquinas 1 y 2 son: 1,4,3,5,2 y la de la tercera máquina: 4,1,3,5,2.

$E_{11} = 2$	$E_{12} = 8$	$E_{13} = 15$	$E_{14} = 25$	$E_{15} = 31$	$Z_{111}=1$	$Z_{251}=1$	$Z_{331}=1$	$Z_{421}=1$	$Z_{541}=1$
$E_{21} = 8$	$E_{22} = 16$	$E_{23} = 26$	$E_{24} = 32$	$E_{25} = 33$	$Z_{112}=1$	$Z_{252}=1$	$Z_{332}=1$	$Z_{422}=1$	$Z_{542}=1$
$E_{31} = 18$	$E_{32} = 28$	$E_{33} = 32$	$E_{34} = 33$	$E_{35} = 34$	$Z_{123}=1$	$Z_{253}=1$	$Z_{333}=1$	$Z_{413}=1$	$Z_{543}=1$

Tabla 8. Valores obtenidos para las variables del modelo TS2 modificado

A partir de los valores obtenidos tras solucionar el problema con el modelo modificado de TS2, se crea el diagrama de Gantt representado en el *Diagrama 7*.

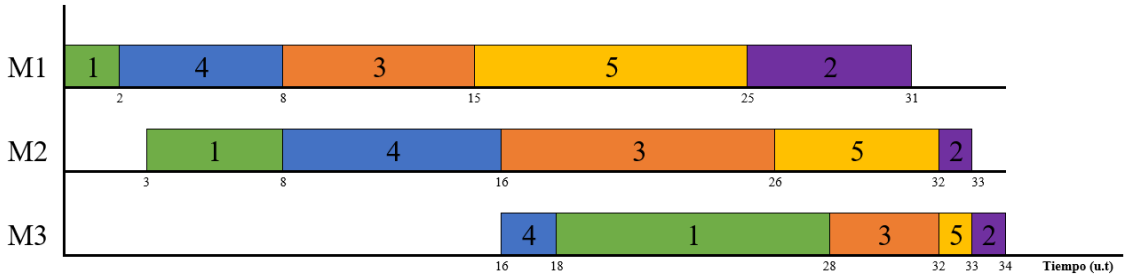


Diagrama 7. Gantt de la solución según el modelo TS2 modificado

3.5.2 Familia de modelos de Manne

Igual que se ha expuesto en los modelos originales, en lo referente a la familia de modelos de Manne, no se utiliza el término "posición del trabajo" por lo que, al trabajar solo con trabajos y máquinas, se simplifica la adaptación de dichos modelos. En definitiva, se prescinde del uso del índice j.

3.5.2.1 Modelo SGST

El modelo SGST solo necesita añadir el subíndice r en la variable D_{ikr} para que se tenga en cuenta que es posible que cada máquina tenga un orden de proceso de los trabajos diferente.

$$Min\ C_{MAX} \tag{4.1}$$

$s.a$:

$$C_{1i} \geq T_{1i} \quad (1 \leq i \leq N) \tag{4.2}$$

$$C_{r+1,i} - C_{ri} \geq T_{r+1,i} \quad (1 \leq r \leq M-1; 1 \leq i \leq N) \tag{4.3}$$

$$C_{ri} - C_{rk} + PD_{ikr} \geq T_{ri} \quad (1 \leq r \leq M; 1 \leq i < k \leq N) \tag{4.4'}$$

$$C_{rk} - C_{ri} + P(1 - D_{ikr}) \geq T_{rk} \quad (1 \leq r \leq M; 1 \leq i < k \leq N) \tag{4.5'}$$

$$C_{MAX} \geq C_{Mi} \quad (1 \leq i \leq N) \tag{4.6}$$

$$C_{ri}, C_{MAX} \geq 0 \qquad (1 \leq r \leq M, 1 \leq i \leq N)$$

$$D_{ikr} \in \{0,1\} \qquad (1 \leq r \leq M, 1 \leq i < k \leq N)$$

Resolución del ejemplo:

En este caso, se exponen dos tipos de variables como resultado de la aplicación del modelo SGST al ejemplo, que se exponen en la *Tabla 9*: la variable C_{ri} que indican el tiempo de finalización del trabajo i en la máquina r, como en el modelo original, y la variable D_{ikr}, que toma valores de 0 o 1 según si se cumple que el trabajo i se sitúe antes del trabajo k en la máquina r.

Y de la misma forma que en la adaptación de los modelos de Wagner, se pueden obtener secuencias diferentes en cada una de las máquinas. En este caso sale para las máquinas 1 y 2 la secuencia: 1,4,3,5,2 y para la máquina 3, la secuencia: 1,3,4,5,2.

$C_{11} = 2$	$C_{12} = 31$	$C_{13} = 15$	$C_{14} = 8$	$C_{15} = 25$	$D_{121} = D_{131} = D_{141} = D_{151} = D_{351} = D_{451} = 1$	$D_{231} = D_{241} = D_{251} = D_{341} = 0$
$C_{21} = 7$	$C_{22} = 33$	$C_{23} = 26$	$C_{24} = 16$	$C_{25} = 32$	$D_{122} = D_{132} = D_{142} = D_{152} = D_{352} = D_{452} = 1$	$D_{232} = D_{242} = D_{252} = D_{342} = 0$
$C_{31} = 26$	$C_{32} = 34$	$C_{33} = 30$	$C_{34} = 32$	$C_{35} = 33$	$D_{123} = D_{133} = D_{143} = D_{153} = D_{343} = D_{353} = D_{453} = 1$	$D_{233} = D_{243} = D_{253} = 0$

Tabla 9. Valores obtenidos para las variables del modelo SGST modificado

Tras el análisis de los resultados obtenidos tras aplicar el algoritmo del modelo SGST modificado al ejemplo, se crea el diagrama de Gantt representado a continuación:

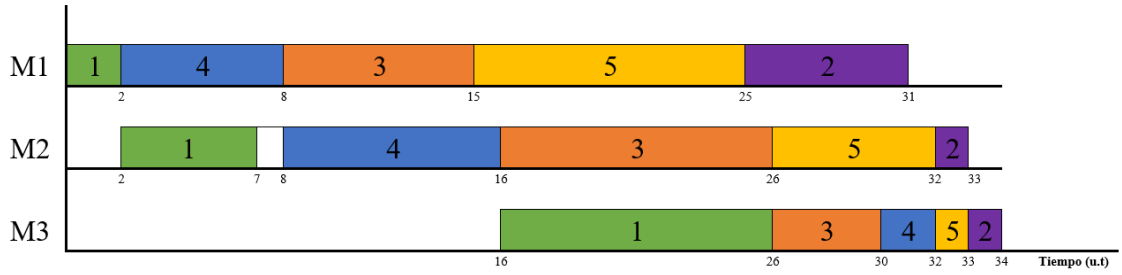


Diagrama 8. Gantt de la solución según el modelo SGST modificado

3.5.2.2 Modelo LYeq

De la misma forma que se ha modificado el modelo SGST se hace el modelo LYeq, puesto que se utilizan las mismas variables con la salvedad de la variable Q_{rik} que ya se encuentra definida en el modelo original LYeq para cada máquina r.

$$Min\ C_{MAX} \tag{5.1}$$

$s.a$:

$$C_{1i} \geq T_{1i} \quad (1 \leq i \leq N) \tag{5.2}$$

$$C_{r+1,i} - C_{ri} \geq T_{r+1,i} \quad (1 \leq r \leq M-1; 1 \leq i \leq N) \tag{5.3}$$

$$C_{MAX} \geq C_{Mi} \quad (1 \leq i \leq N) \tag{5.4}$$

$$PD_{ikr} + C_{ri} - C_{rk} - T_{ri} = Q_{rik} \quad (1 \leq r \leq M; 1 \leq i < k \leq N) \tag{5.5'}$$

$$Q_{rik} \leq P - T_{ri} - T_{rk} \quad (1 \leq r \leq M; 1 \leq i < k \leq N) \tag{5.6}$$

$$C_{ri}, C_{MAX}, Q_{rik} \geq 0 \qquad (1 \leq r \leq M, 1 \leq i < k \leq N)$$

$$D_{ikr} \in \{0,1\} \qquad (1 \leq r \leq M, 1 \leq i < k \leq N)$$

Resolución del ejemplo:

El modelo LYeq cuenta con las mismas variables que el modelo SGST para definirse. Los resultados se muestran en la *Tabla 10*.

Al igual que en el caso de la resolución según el modelo SGST, gracias a la variable D_{ikr} se obtiene la secuencia para cada máquina.

Se obtiene la secuencia: 1,4,3,5,2 para las máquinas 1 y 2, y la secuencia: 4,1,3,5,2 para la tercera máquina.

$C_{11} = 2$	$C_{12} = 31$	$C_{13} = 15$	$C_{14} = 8$	$C_{15} = 25$	$D_{121} = D_{131} = D_{141} = D_{151} = D_{351} = D_{451} = 1$	$D_{231} = D_{241} = D_{251} = D_{341} = 0$
$C_{21} = 7$	$C_{22} = 33$	$C_{23} = 26$	$C_{24} = 16$	$C_{25} = 32$	$D_{122} = D_{132} = D_{142} = D_{152} = D_{352} = D_{452} = 1$	$D_{232} = D_{242} = D_{252} = D_{342} = 0$
$C_{31} = 28$	$C_{32} = 34$	$C_{33} = 32$	$C_{34} = 18$	$C_{35} = 33$	$D_{123} = D_{133} = D_{153} = D_{353} = D_{453} = 1$	$D_{143} = D_{233} = D_{243} = D_{253} = D_{343} = 0$

Tabla 10. Valores obtenidos para las variables del modelo LYeq modificado

Se obtiene entonces el Diagrama de Gantt gracias a los resultados obtenidos al resolver el problema con el modelo modificado de LYeq mostrado en el *Diagrama 9*.

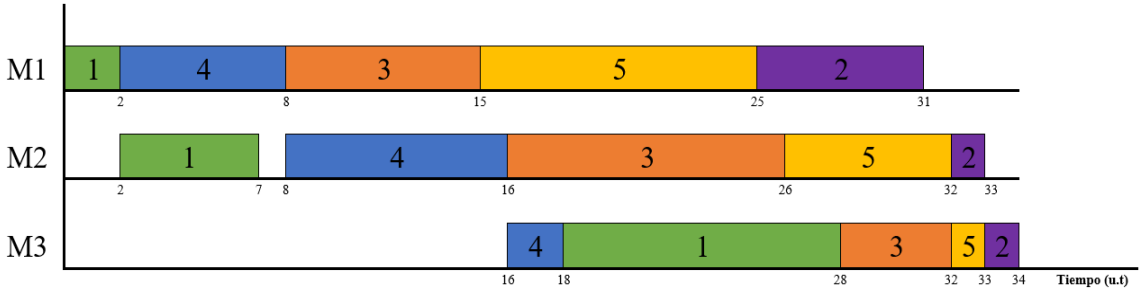


Diagrama 9. Gantt de la solución según el modelo LYeq modificado

4 RESOLUCIÓN Y ANÁLISIS DE LOS RESULTADOS

4.1 Proceso de resolución

Tras la modificación de los modelos originales, se dispone de cuatro modelos adaptados al entorno de *Flowshop sin permutación*, problema propuesto para este proyecto.

Cada modelo resolverá una batería de 160 instancias o problemas. Los problemas se diferencian por el número de máquinas y el número de trabajos. El número de máquinas se genera de 2 a 5, y el número de trabajos será 5, 10, 15 o 20. De cada una de las combinaciones se tienen, además, 10 instancias diferentes.

La resolución de la batería de instancias llevará a la ejecución de un total de $4[N]\ x\ 4[M]\ x\ 10[inst]\ x\ 4[MOD] = 640$ problemas diferentes mediante el solver Gurobi. Gurobi es un solver comercial de optimización de programación lineal (Optimization, 2018).

Se aclara lo que se tiene y lo que se necesita para ejecutar el solver de Gurobi:

- En primer lugar, se dispone de: la definición de los modelos en forma de algoritmo además de una batería de instancias en archivos de texto con sus datos. Igual que se muestra en la Figura 2.
- Se necesita obtener un archivo de extensión $.lp$ para cada una de las instancias formuladas y para cada modelo. Este archivo debe tener una sintaxis determinada y tendrá una apariencia del estilo de la Figura 3.

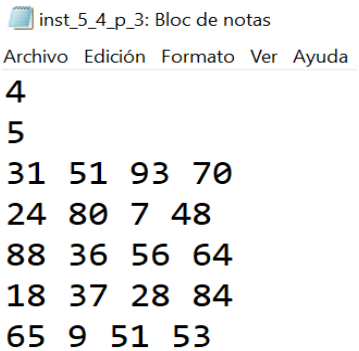


Figura 2. Disposición de los datos de la instancia número 3 para M=5 y N=4

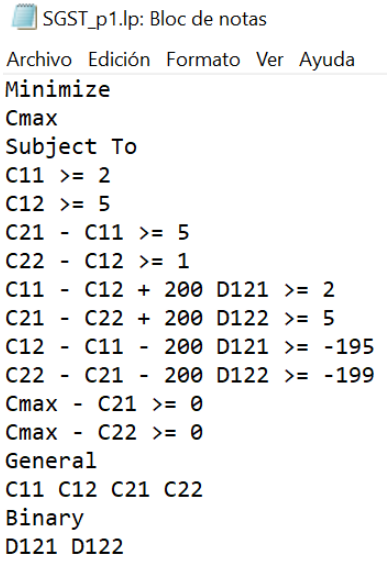


Figura 3. Sintaxis del modelo SGST de un ejemplo de M=2 y N=2

Para obtener el archivo ejecutable por Gurobi se ha escrito en leguaje C un programa que automatiza todas las acciones.

- Primero se generan varios archivos auxiliares donde se almacenan los nombres de los archivos de

extensión . lp según modelos.

- Luego se escriben esos archivos de tipo . lp de acuerdo con el algoritmo que se ha definido para cada modelo, tomando los datos de la batería de instancias.
- Se genera un archivo de texto para cada modelo que contiene, por líneas, las órdenes de ejecución que se le mandan a Gurobi como se puede observar en la *Figura 4*. Además, en la orden que se le da a Gurobi se le ha puesto una restricción en el tiempo de resolución. Esta orden se implementa con *TimeLimit=900* y significa que, si no se llega a la solución óptima en 900 segundos, debe parar la resolución y devolver el valor de la mejor solución factible encontrada para la función objetivo definida.
- Por último, estos archivos de texto se cambian de extensión para su ejecución. Pasan a ser archivos con extensión . bat

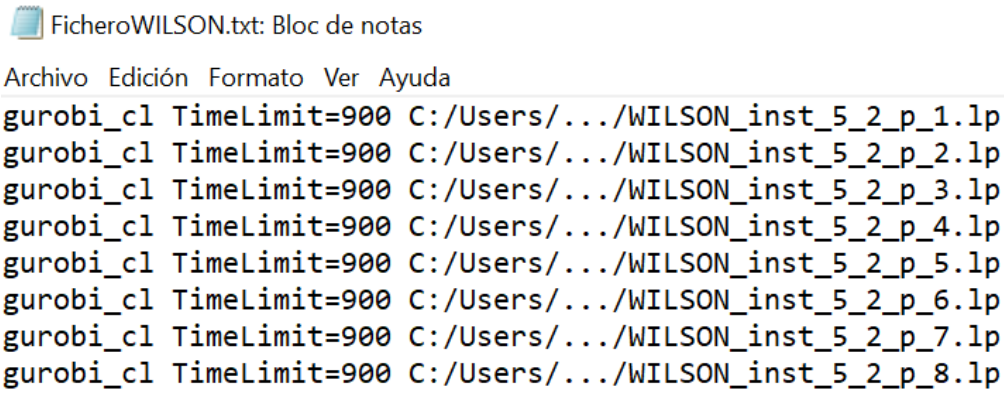


Figura 4. Visualización del archivo de texto para ejecutar las órdenes en Gurobi

Se procede ahora modelo a modelo de igual forma. La ejecución de los archivos de extensión . bat ejecuta Gurobi y da como resultado un archivo de extensión . log, el cual, con ayuda de un ejecutable, crea un archivo de extensión . csv. En este último archivo se tienen los resultados de la simulación: el valor del tiempo de cómputo (*CPU Time*) y el valor de la mejor solución encontrada, de la forma en que se muestra en la *Figura 5*, de todas las instancias de dicho modelo.

C:/Users/.../WILSON_inst_5_2_p_1.lp	0,03	2,63E+02
C:/Users/.../WILSON_inst_5_2_p_2.lp	0,12	3,06E+02
C:/Users/.../WILSON_inst_5_2_p_3.lp	0,03	3,68E+02
C:/Users/.../WILSON_inst_5_2_p_4.lp	0,12	4,07E+02
C:/Users/.../WILSON_inst_5_2_p_5.lp	0,14	3,45E+02
C:/Users/.../WILSON_inst_5_2_p_6.lp	0,05	2,92E+02
C:/Users/.../WILSON_inst_5_2_p_7.lp	0,11	2,80E+02
C:/Users/.../WILSON_inst_5_2_p_8.lp	0,34	2,76E+02

Figura 5. Visualización del archivo Excel de los resultados obtenidos con Gurobi

Una vez ordenados y agrupados los resultados obtenidos en un archivo de Excel, se procede con un análisis comparativo de las soluciones según el modelo utilizado, según el tiempo de cómputo, etc. Para llevarlo a cabo

se utilizará el programa SPSS, software estadístico de IBM.

4.2 Análisis de los resultados

Se realiza en este apartado el análisis de los resultados obtenidos. Se ha llevado a cabo la resolución de un total de 640 instancias. En definitiva, se pretende determinar qué modelo es el más eficiente para resolver las instancias del problema propuesto en función de diferentes variables o parámetros.

En cuanto a la obtención de soluciones, se va a analizar la existencia de resultados factibles y no factibles según el modelo por el cual se ha resuelto. También se comprueba si en el caso de tener diferentes soluciones factibles, se ha encontrado la solución óptima.

Resulta interesante estudiar la diferencia entre la mejor solución encontrada y la solución hallada con cada modelo, lo cual se hace con el indicador ARPD. Se puede comprobar, mediante un análisis de la varianza, la dependencia que tiene esta variable ARPD con los factores que cambian en el problema como son el modelo utilizado para la resolución, el número de máquinas (M) o el número de trabajos (N).

Por otro lado, otra variable a la que se le podría dar cierta importancia es al tiempo de cómputo. Sin embargo, en este caso no se va a detallar demasiado puesto que se ha impuesto un tiempo máximo de resolución, 900s o 15 min, y muchas de las soluciones se han encontrado en torno a este tiempo, por lo que las medidas quedan perturbadas con estos valores tan altos.

4.2.1 Soluciones factibles y no factibles

La definición de solución factible es aquella solución obtenida tras la resolución de un problema mediante cualquiera de los modelos comentados anteriormente. Esta solución puede ser óptima o no, pero válida al cumplir las restricciones definidas por el modelo en cuestión.

Las soluciones no factibles se dan cuando los modelos de programación no son capaces de encontrar una solución válida para el problema. Esto se da con más probabilidad en modelos cuyas restricciones sean del tipo menor o igual.

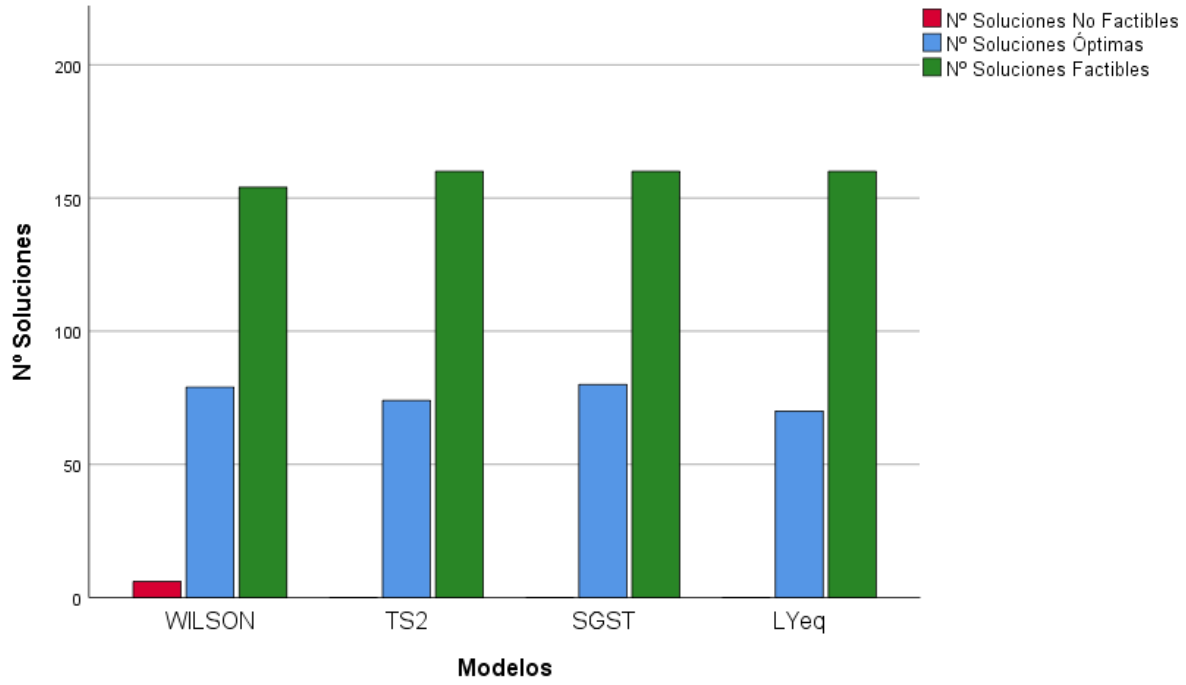
Las soluciones óptimas en este caso se van a tomar como las soluciones que se hayan encontrado en un tiempo inferior a 900 segundos. Independientemente de ser el menor valor de la función objetivo de entre las soluciones obtenidas.

En la *Tabla 11* se muestra el resumen del número de soluciones factibles, soluciones no factibles y soluciones óptimas agrupados por modelo de resolución.

Como se puede observar también en la *Gráfica 1*, el modelo WILSON es el único modelo que obtiene soluciones no factibles, no es capaz de resolver 6 instancias de la batería. En cuanto al número de soluciones óptimas de cada modelo, se observa que giran en torno al 50% de las soluciones factibles, siendo el modelo LYeq el modelo con el que menos soluciones óptimas se obtienen.

Modelos	Nº Simulaciones	Nº Soluciones Factibles	Nº Soluciones No Factibles	Nº Soluciones Óptimas
WILSON	160	154	6	79
TS2	160	160	0	74
SGST	160	160	0	80
LYeq	160	160	0	70

Tabla 11. Resumen de los resultados de las simulaciones según modelo



Gráfica 1. Representación del número de soluciones halladas según modelos

4.2.2 Análisis ARPD

Es importante destacar que, aunque también se pueden utilizar los valores de la función objetivo obtenidos como indicadores a estudiar, es interesante emplear la medida del porcentaje de desviación relativa, del término inglés *Relative Percentage Deviation* (*RPD*), de cada algoritmo e instancia. El RPD de una instancia l y un algoritmo a con respecto al de *referencia*, se define como:

$$RPD(l) = RPD_{a,ref}(l) = \frac{OFV_a(l) - OFV_{ref}(l)}{OFV_{ref}(l)}$$

También se indica a continuación cómo se calcula el indicador ARPD aplicado a un algoritmo. ARPD son las siglas del término inglés *Average Relative Percentage Deviation*. Si se considera una batería de un total de L instancias, el ARPD de un algoritmo se define según:

$$ARPD = ARPD_{a,ref} = \frac{1}{L}\sum_{l=1}^{L} \frac{OFV_a(l) - OFV_{ref}(l)}{OFV_{ref}(l)}$$

Siendo $OFV_a(l)$ el valor obtenido para la función objetivo por un determinado algoritmo y $OFV_{ref}(l)$ el valor obtenido para la función objetivo por el algoritmo de referencia, normalmente mejor solución obtenida para la instancia.

Como se explica en la literatura (Framinan, Leisten and Ruiz García, 2014), en referencias anteriores solo se utilizaba el análisis de la desviación estándar o de los valores de la varianza de ARPD cuando se debatía sobre el rendimiento de los algoritmos. Esto se debe a que el ARPD se refiere a un rendimiento medio cuando normalmente, los algoritmos de programación funcionan de manera diferente dependiendo del tipo de instancia que evalúe. Sin embargo, empieza a ser común el uso de test estadísticos para evaluar comportamientos relativos de métodos de resolución aproximados. Aprueban estas aproximaciones estadísticas ya que dan, al menos, algún razonamiento estadístico para comportamientos relativos de las aproximaciones en cuestión.

En general, $OFV_{ref}(l)$ sería el valor de la solución óptima y, por tanto, el ARPD indicaría la desviación promedio que hay con respecto al valor óptimo de la función objetivo. Sin embargo, para problemas más

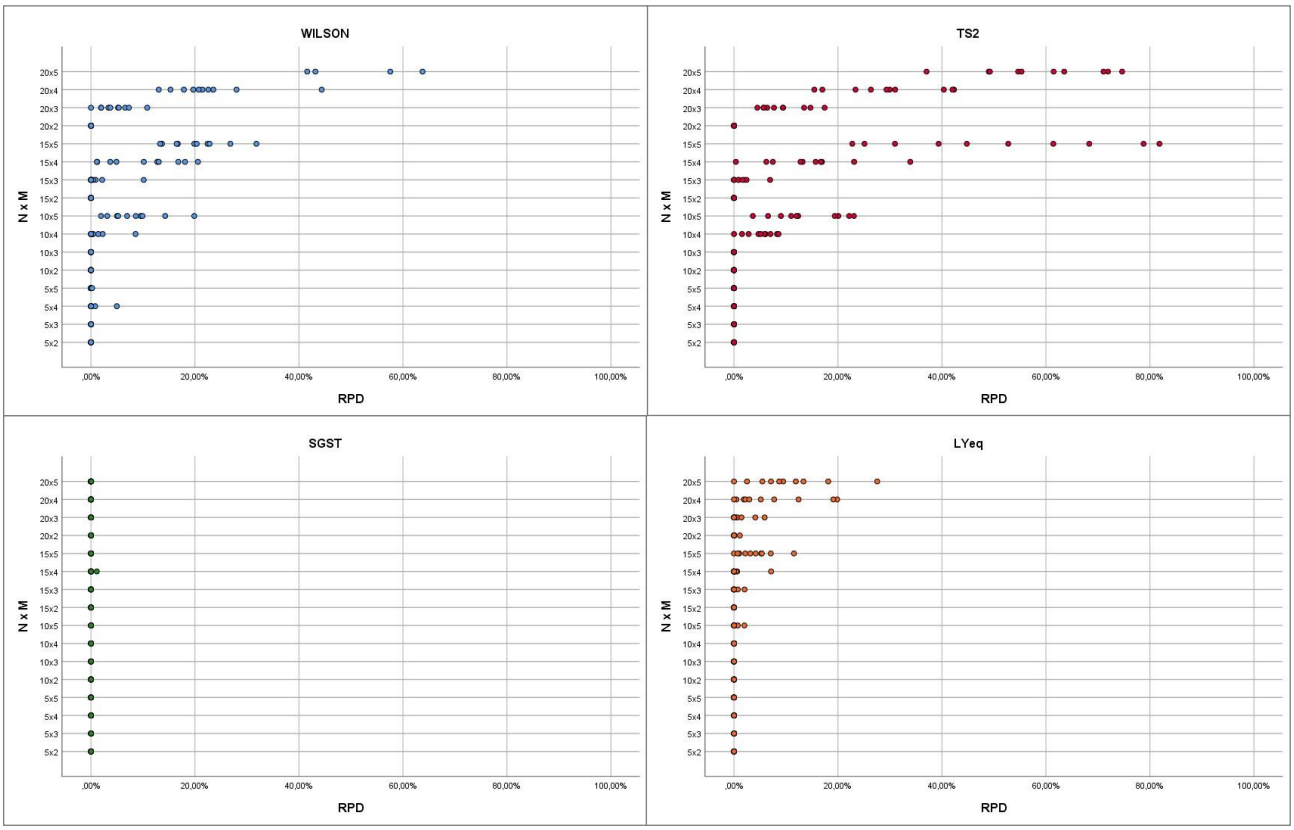
complejos, un algoritmo no puede compararse con la solución óptima, sino que se compara con otro algoritmo u otro conjunto de ellos.

Para este caso en concreto se equipara la solución óptima de la función objetivo a la mejor solución de entre las encontradas aplicando los diferentes modelos. Gracias a esto se puede también asemejar la hipótesis para aplicar el procedimiento de la evaluación estadística. Siendo MOD_i y MOD_j los algoritmos o modelos trabajados quedaría:

$$Hipótesis: ARPD_{MOD_i,ref} \leq ARPD_{MOD_j,ref}$$

El valor de referencia $OFV_{ref}(l)$ se toma entonces como el mejor valor, el mínimo valor de la función objetivo entre los obtenidos con los diferentes modelos para cada una de las instancias, es decir: $OFV_{ref} = min\,(OFV_{MOD_1}, OFV_{MOD_2}, OFV_{MOD_3}, OFV_{MOD_4})$. La confirmación de la hipótesis propondrá que el MOD_i funciona mejor que el MOD_j con respecto al valor de referencia y sujeto a un nivel de confianza.

Explicada la teoría, se expone a continuación la representación de todos los valores RPD según modelo y con respecto a NxM en la *Gráfica 2*.



Gráfica 2. Dispersión de los valores RPD con respecto a NxM

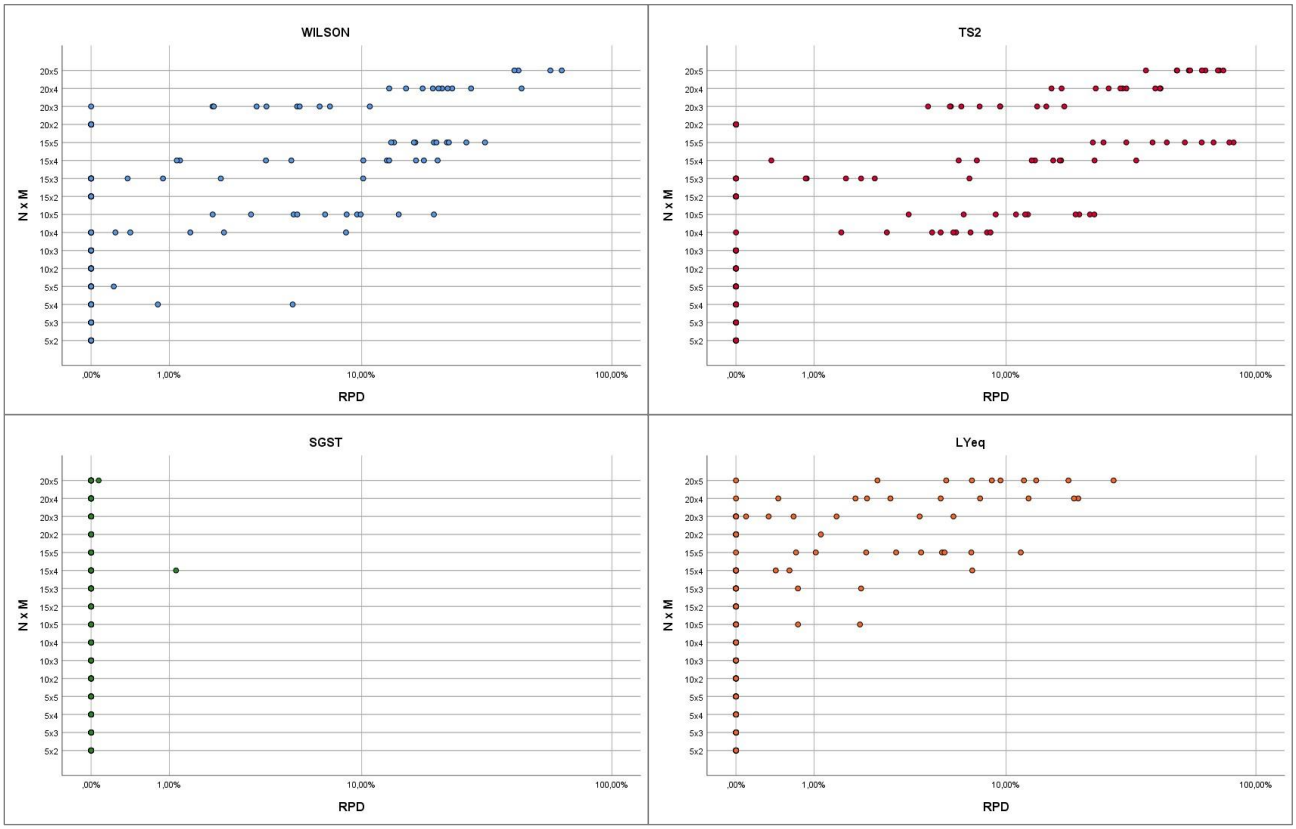
Como se puede observar en la gráfica de dispersión:

- Para los problemas más sencillos, como son los de cinco trabajos e incluso los de diez trabajos, casi todas las soluciones encontradas para la función objetivo por los cuatro modelos son las mismas.
- Al aumentar el número de trabajos N, se aprecia un aumento en la dispersión de los valores de RPD. Es decir, cuando aumenta la complejidad en cuanto al número de trabajos en las instancias resueltas, se obtienen diferentes resultados según el modelo que se utilice para la resolución.
- Conforme al número de máquinas M, es sobre todo en los valores de cuatro y cinco donde se encuentran los mayores despuntes. En estos casos, el problema es bastante complejo y según el

modelo que se utilice, se obtienen resultados más o menos distantes con respecto a la mejor solución.

- En general, para los problemas menos complejos, siendo dos o tres máquinas y cualquier número de trabajos, el porcentaje de desviación se encuentra dentro del 20%.
- En cuanto a los modelos, el modelo SGST tiene desviación nula para prácticamente todas las instancias. Esto quiere decir que ha sido ese modelo el que ha encontrado la mejor solución. Por el contrario, es el modelo TS2 el que presenta mayores valores de desviación de la mejor solución.

Con el fin de analizar de forma más exhaustiva el rango de 0 a 10% de RPD, se representan los mismos valores de la gráfica anterior con el eje x en escala logarítmica en la *Gráfica 3*.



Gráfica 3. Dispersión RPD - NxM, en escala logarítmica

Del grupo de gráficas anteriores se pueden obtener las siguientes conclusiones:

- En todos los problemas considerados se observa lo siguiente: el modelo SGST obtiene los resultados más próximos al mejor obtenido que se coge como referencia para calcular el RPD. Todos los puntos están en el rango de dispersión de 0 a 1%.
- En los problemas más complejos, donde hay mayor pico de dispersión, se observa el siguiente orden en el sentido creciente del eje x: los resultados obtenidos con el modelo LYeq son los primeros, después se aprecia mayor concentración de resultados procedentes del modelo WILSON y los últimos se refieren a los resultados obtenidos con el modelo TS2.

4.2.2.1 ARPD con respecto al modelo

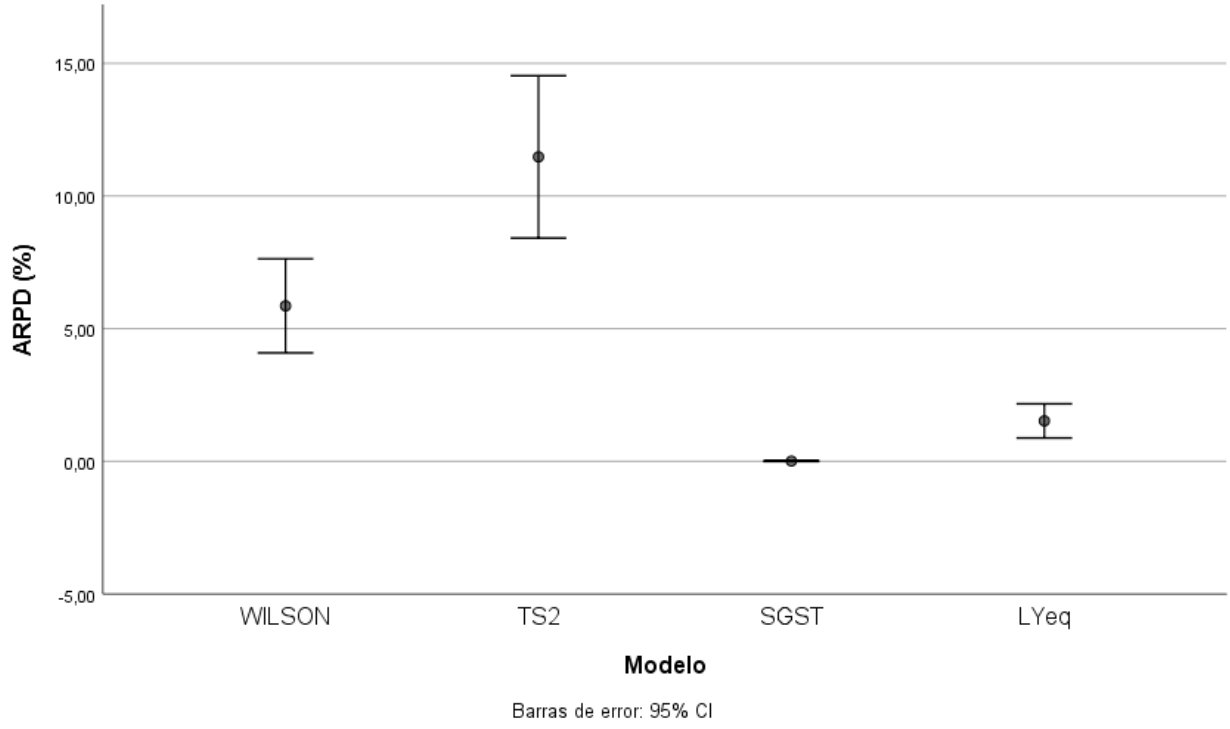
Ahora se hace una comparación de forma genérica de las medias de los valores de ARPD obtenidos según el modelo utilizado para la resolución de las instancias. La *Tabla 12* resume los valores representados en la *Gráfica 4*.

Modelo	Media	N	Desviación
WILSON	5,8583%	154	11,13668%
TS2	11,4746%	160	19,58243%
SGST	0,0074%	160	0,08886%
LYeq	1,5229%	160	4,11588%

Tabla 12. Resultados obtenidos de la media ARPD respecto a los modelos

Para evaluar cómo afecta el hecho de resolver las mismas instancias con los diferentes modelos, se utilizan la media de ARPD y la desviación estándar de dicho valor.

El valor que aparece como *N* en la tabla es el número de instancias consideradas. Aunque anteriormente se ha indicado que existían 160 instancias a resolver para cada modelo, hay 6 instancias que no son resueltas con el modelo WILSON, dichas soluciones son las definidas como no factibles en el apartado anterior.



Gráfica 4. Medias de los valores ARPD respecto a los modelos

De la gráfica anterior se pueden obtener las siguientes conclusiones:

- Tomando los valores de las medias, haciendo una división por familias de modelos se puede decir que la familia de modelos de Wagner tiene las medias de los valores de ARPD mayores que las medias obtenidas para la familia de modelos de Manne, en torno a un 10%.
- En lo que respecta al nivel de confianza, tomado como el 95%, se puede observar que ninguna barra de error se superpone entre los modelos. Además, es interesante analizar el valor de esta desviación, para los modelos de Wagner el valor es significativamente mayor que para los modelos

de la familia de Manne. Esto se traduce en la existencia de un mayor rango de valores obtenidos para los modelos de la familia de Wagner.

De este análisis se puede concluir que los modelos que dan mejor resultado al resolver las instancias del problema serían los modelos de la familia de Manne ya que obtienen el mejor valor de entre los encontrados, o para expresarlo de otra forma, el valor óptimo de la función objetivo o uno muy próximo a este.

Tras esta conclusión, se puede plantear otro enfoque: hacer desglose de los valores obtenidos según el número de máquinas y según el número de trabajos, para cada modelo. Se procede a la explicación de los resultados de dicho análisis.

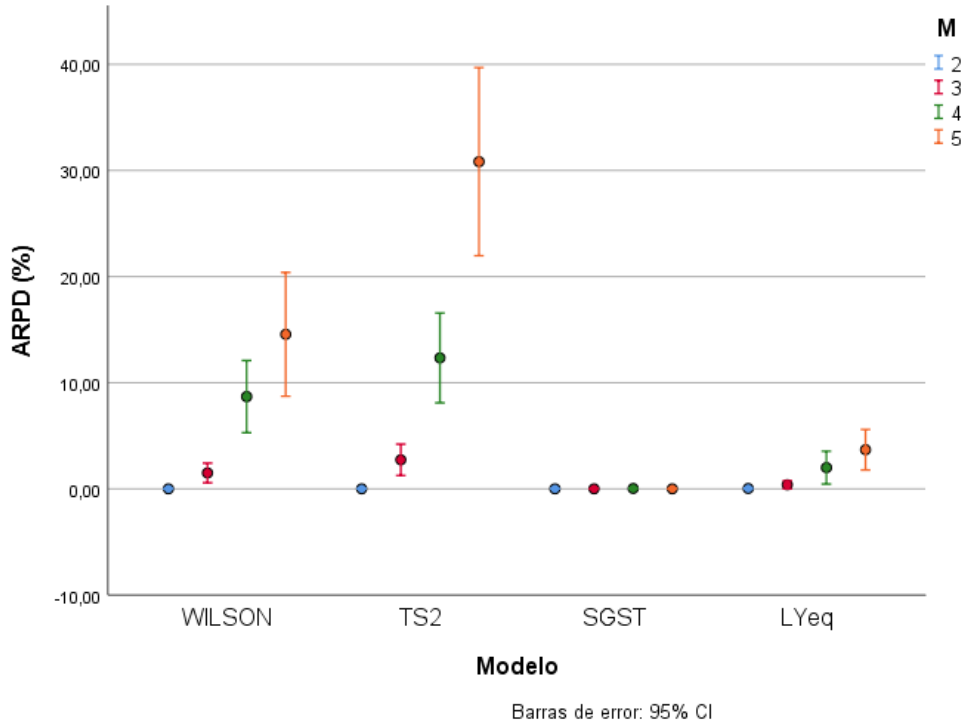
4.2.2.2 ARPD con respecto al número de máquinas

Los valores obtenidos son recogidos en la *Tabla 13*. En esta tabla se muestra la media de cada modelo respecto al número de máquinas, además de la desviación estándar y el número de instancias consideradas *N*.

M		WILSON	TS2	SGST	LYeq
2	**Media**	0,0000%	0,0000%	0,0000%	0,0281%
	N	40	40	40	40
	Desviación	0,00000%	0,00000%	0,00000%	0,17781%
3	**Media**	1,4948%	2,7318%	0,0000%	0,3826%
	N	40	40	40	40
	Desviación	2,86957%	4,61139%	0,00000%	1,16601%
4	**Media**	8,6868%	12,3387%	0,0281%	1,9977%
	N	40	40	40	40
	Desviación	10,63408%	13,23340%	0,17746%	4,82671%
5	**Media**	14,5563%	30,8281%	0,0017%	3,6832%
	N	34	40	40	40
	Desviación	16,71314%	27,72418%	0,01096%	5,97777%

Tabla 13. Resultados obtenidos de la media ARPD con respecto a los modelos y agrupados según M

Los valores de la tabla anterior son representados en la *Gráfica 5* con el software SPSS. Con ayuda del código de colores se observan fácilmente los diferentes resultados para cada número de máquinas:



Gráfica 5. Medias de los valores ARPD con respecto a los modelos y agrupados según M

De los resultados representados en la *Gráfica 5* se puede obtener la información siguiente:

- Para valores de $M = 2$, representado en azul en la gráfica, los resultados obtenidos por todos los modelos coinciden entre ellos y con el mejor resultado de la función objetivo hallado.
- Para valores de $M = 3$ la desviación de los resultados en los modelos de la familia de Wagner es mínima, pero algo mayor que en el caso de la familia de modelos de Manne, que es nula.
- En el caso de la existencia de 4 máquinas en el problema, representado con el color verde en la gráfica, la desviación con respecto a la mejor solución encontrada se incrementa. Para los modelos de WILSON y TS2, la media de la desviación gira en torno al 10% con una desviación típica también de aproximadamente un 10%. Sin embargo, el modelo SGST se mantiene con una desviación prácticamente nula y el modelo LYeq con una desviación también insignificante.
- Por último, para valores de $M = 5$, los problemas considerados más complejos, se obtiene lo siguiente: en los dos primeros modelos, WILSON y TS2, se tiene una media de la desviación que supera el 20% y una desviación típica bastante considerable en comparación a las demás. Por su parte, los modelos SGST y LYeq se mantienen en la misma línea que para el caso anterior.

De este análisis se puede sacar una conclusión clara, el modelo que mejor funciona, en términos de encontrar la mejor solución, sería el modelo SGST que obtiene, para todos los valores de M, una media de desviación casi nula con respecto a la mejor solución.

4.2.2.3 ARPD con respecto al número de trabajos

Se procede de la misma forma que se ha hecho anteriormente: se exponen los resultados obtenidos en la *Tabla 14*, agrupados por modelo y según el número de trabajos. Para mostrarlo de forma visual se hace uso de la representación en la *Gráfica 6*.

N		WILSON	TS2	SGST	LYeq
5	**Media**	0,1499%	0,0000%	0,0000%	0,0000%
	N	40	40	40	40
	Desviación	0,79224%	0,00000%	0,00000%	0,00000%
10	**Media**	2,4337%	4,7316%	0,0000%	0,0685%
	N	40	40	40	40
	Desviación	4,58536%	6,76007%	0,00000%	0,33486%
15	**Media**	8,0017%	16,6807%	0,0281%	1,2841%
	N	40	40	40	40
	Desviación	9,51467%	23,55813%	0,17746%	2,59644%
20	**Media**	14,0814%	24,4862%	0,0017%	4,7389%
	N	34	40	40	40
	Desviación	17,70590%	24,07453%	0,01096%	6,85542%

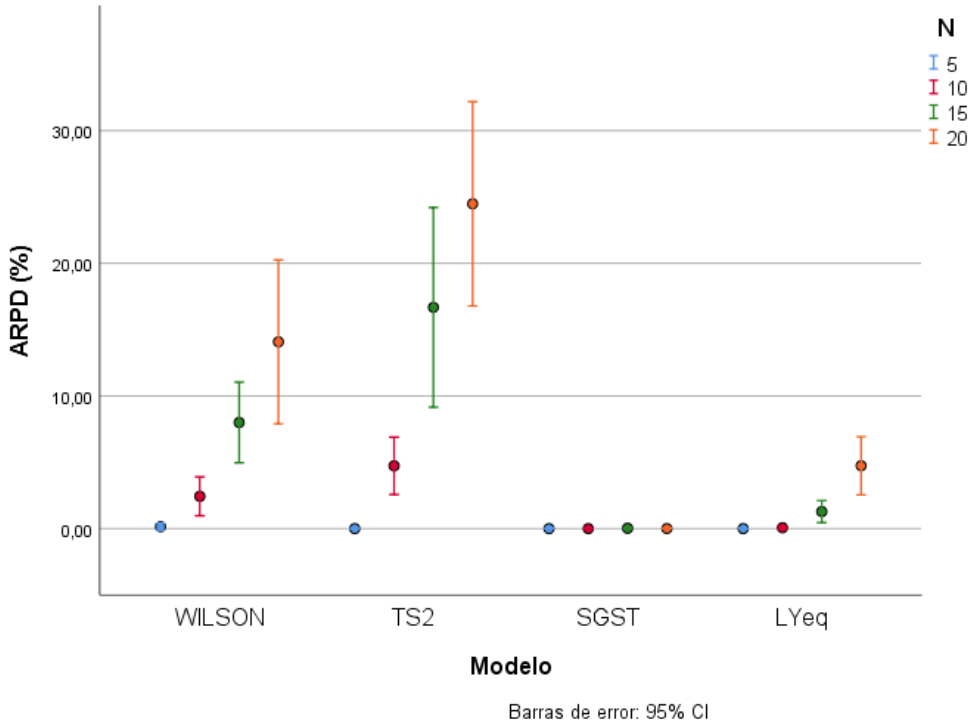
Tabla 14. Resultados obtenidos de la media ARPD con respecto a los modelos y agrupados según N

En este caso, en la tabla se exponen los resultados divididos por modelos y para cada uno de los valores de N (5, 10, 15 y 20) se obtienen los valores de la media de ARPD, la desviación típica de este valor y N, el número de instancias tomadas para dicho cálculo.

Los resultados obtenidos en cuanto a los modelos son parecidos a los obtenidos hasta el momento:

- Para los modelos WILSON y TS2 se observa que, según el número de trabajos aumenta, la media del porcentaje de ARPD también lo hace, y lo que es más significativo, la desviación típica es significativamente mayor. Esto supone que para dichos modelos se obtienen resultados en un amplio rango.
- En el caso de los modelos de la familia de Manne los valores de la media de dispersión son muy cercanos a cero y la desviación típica es muy pequeña en casi todos los casos. Cabe destacar el modelo SGST, que resuelve los problemas para todo valor de N de forma que encuentra siempre el mejor valor de la función objetivo.

En conclusión, se llega a la misma deducción, el modelo SGST sería el indicado si se quiere obtener con seguridad los mejores valores para la función objetivo. En segundo lugar, se puede aconsejar el uso del modelo LYeq que obtiene resultados comparables al modelo anterior.



Gráfica 6. Medias de los valores ARPD con respecto a los modelos y agrupados según N

4.2.3 Análisis de la varianza, ANOVA

A continuación, se plantean una serie de preguntas que se podrían predecir sin necesidad de la resolución del análisis de la varianza. Sin embargo, con ayuda de este análisis se confirmará la dependencia de los valores obtenidos de ARPD con respecto de las variables que cambian en la resolución de los problemas: el modelo utilizado, el número de máquinas M y el número de trabajos N.

- ¿Afecta el modelo utilizado para resolver los problemas al valor de ARPD?
- ¿Afecta el número de máquinas de los problemas al valor de ARPD?
- ¿Afecta el número de trabajos de los problemas al valor de ARPD?

Para cada una de las preguntas anteriores se utiliza un contraste de hipótesis. Se necesita proponer una hipótesis nula y una hipótesis alternativa. En el primer caso se va a explicar detalladamente el procedimiento seguido y para los siguientes casos se seguirá un proceso análogo.

4.2.3.1 Dependencia con el modelo

En el caso de la dependencia con el modelo utilizado se propone lo siguiente:

$$\mu_{MOD1}: media\ de\ valores\ ARPD\ para\ el\ modelo\ WILSON$$
$$\mu_{MOD2}: media\ de\ valores\ ARPD\ para\ el\ modelo\ TS2$$
$$\mu_{MOD3}: media\ de\ valores\ ARPD\ para\ el\ modelo\ SGST$$
$$\mu_{MOD4}: media\ de\ valores\ ARPD\ para\ el\ modelo\ LYeq$$

$$H_0: \mu_{MOD1} = \mu_{MOD2} = \mu_{MOD3} = \mu_{MOD4}\ \ el\ modelo\ no\ afecta\ a\ los\ valores\ de\ ARPD\ obtenidos$$
$$H_1: \mu_{MODi} \neq \mu_{MODj}\ para\ i \neq j\ \ \ el\ modelo\ afecta\ a\ los\ valores\ de\ ARPD\ obtenidos$$

Con ayuda del programa SPSS y haciendo uso de la herramienta análisis de la ANOVA de un factor se obtienen

los valores de la siguiente tabla:

	Suma de cuadrados	gl	Media cuadrática	F	Sig.
Entre grupos	12688,166	3	4229,389	32,241	**,000**
Dentro de grupos	82642,677	630	131,179		
Total	95330,843	633			

Tabla 15. ANOVA con dependencia del modelo

Para aceptar o rechazar la hipótesis expuesta anteriormente se analiza el p-valor. El nivel de error del contraste que se toma por defecto es del 5% ($\alpha = 0{,}05$). Se comprueba entonces que el p-valor, marcado en negrita en la tabla, es menor que el nivel de significación definido (o alfa).

La muestra presenta suficiente evidencia como para rechazar la hipótesis nula de que las medias de los valores sean iguales para todos los modelos. Se llega a la conclusión de que el modelo utilizado al resolver los problemas influye en los valores obtenidos de ARPD.

4.2.3.2 Dependencia con el número de máquinas

De la misma forma, se procede para el caso de la dependencia con el número de máquinas:

$$\mu_{M2}: media\ de\ valores\ ARPD\ para\ M = 2$$
$$\mu_{M3}: media\ de\ valores\ ARPD\ para\ M = 3$$
$$\mu_{M4}: media\ de\ valores\ ARPD\ para\ M = 4$$
$$\mu_{M5}: media\ de\ valores\ ARPD\ para\ M = 5$$

$H_0: \mu_{M2} = \mu_{M3} = \mu_{M4} = \mu_{M5}$, *el número de máquinas no afecta a los valores de ARPD obtenidos*

$H_1: \mu_{Mi} \neq \mu_{Mj}\ para\ i \neq j,$ *el número de máquinas afecta a los valores de ARPD obtenidos*

Se obtienen ahora los siguientes resultados:

	Suma de cuadrados	gl	Media cuadrática	F	Sig.
Entre grupos	14330,396	3	4776,799	37,153	**,000**
Dentro de grupos	81000,447	630	128,572		
Total	95330,843	633			

Tabla 16. ANOVA con dependencia del número de máquinas M

De este resultado también se puede confirmar que el número de máquinas afecta en los valores de ARPD.

4.2.3.3 Dependencia con el número de trabajos

Por último, se estudia la dependencia con el número de trabajos del problema y se propone:

$$\mu_{N5}: media\ de\ valores\ ARPD\ para\ N = 5$$
$$\mu_{N10}: media\ de\ valores\ ARPD\ para\ N = 10$$
$$\mu_{N15}: media\ de\ valores\ ARPD\ para\ N = 15$$
$$\mu_{N20}: media\ de\ valores\ ARPD\ para\ N = 20$$

$H_0: \mu_{N5} = \mu_{N10} = \mu_{N15} = \mu_{N20}$, *el número de trabajos no afecta a los valores de ARPD obtenidos*

$H_1: \mu_{Ni} \neq \mu_{Nj}\ para\ i \neq j$, *el número de trabajos afecta a los valores de ARPD obtenidos*

Tras ejecutar el análisis de la varianza se obtiene para este caso:

	Suma de cuadrados	gl	Media cuadrática	F	Sig.
Entre grupos	10878,100	3	3626,033	27,049	**,000**
Dentro de grupos	84452,743	630	134,052		
Total	95330,843	633			

Tabla 17. ANOVA con dependencia del número de trabajos N

Se obtiene un p-valor inferior al valor de alfa definido, por tanto, también se confirma que el número de trabajos del problema afecta a los valores obtenidos de ARPD.

En resumen, este análisis confirma la dependencia de la variable ARPD con el número de máquinas y el número de trabajos, así como con el modelo de resolución utilizado.

4.2.4 Análisis CPU Time

Como se ha comentado anteriormente, el análisis del tiempo de cómputo se puede ver perturbado por la restricción del límite de tiempo de resolución. Este límite está en 900 segundos, y muchas de las soluciones se encuentran en torno a ese instante.

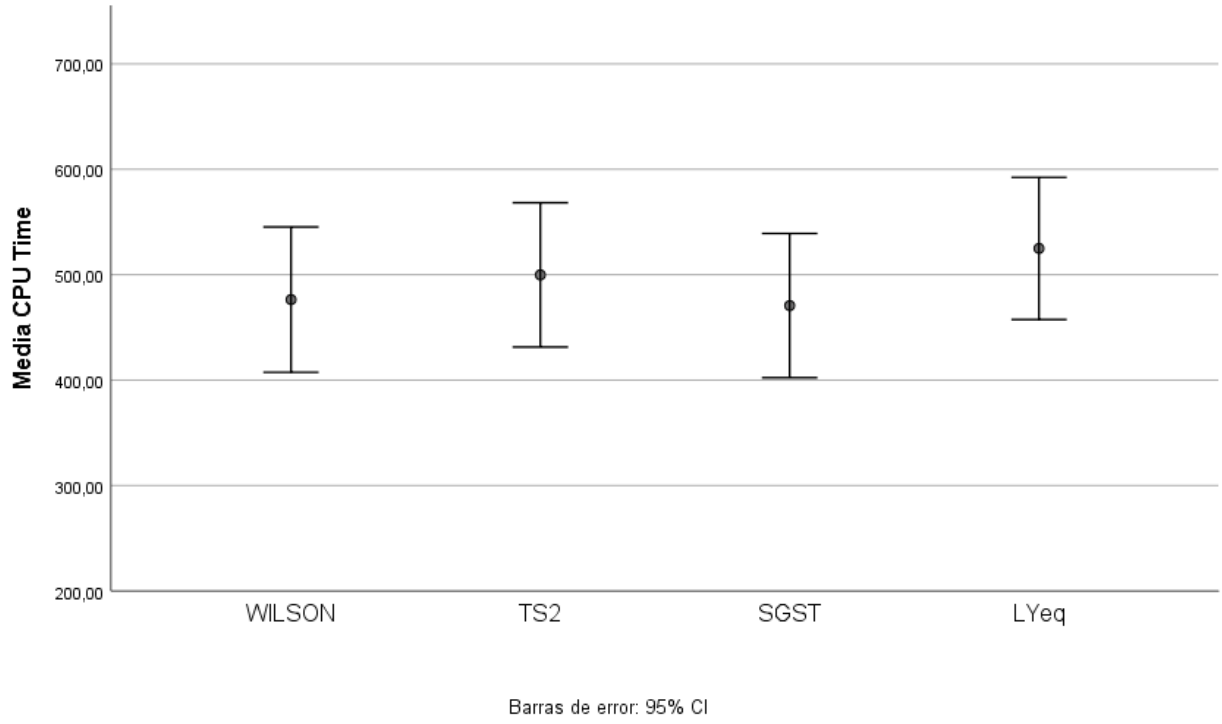
4.2.4.1 CPU Time con respecto al modelo

Se muestra en la *Tabla 18* la media de CPU Time total de todos los casos, sin diferenciar por número de máquinas M o número de trabajos N. Además, se indica el valor de la desviación estándar de esta medida.

Modelo	Media	N	Desv. Estándar
WILSON	476,371625	160	441,5528949
TS2	499,78975	160	438,6533359
SGST	470,6953125	160	438,2783187
LYeq	524,907125	160	431,9190949

Tabla 18. Resultados obtenidos de la media de CPU Time con respecto a los modelos

En la *Gráfica 7* se puede observar la representación del valor medio del CPU Time con respecto a los diferentes modelos, con un intervalo de confianza del 95%.



Gráfica 7. Medias del CPU Time con respecto a los modelos

De dichos resultados se pueden sacar algunas conclusiones:

- Las medias de los tiempos de cómputo se encuentran en torno a los 500s. El modelo que menos tarda, de media, en la resolución de las instancias es el modelo SGST. El modelo que más tarda sería el modelo LYeq.
- Las desviaciones estándar con respecto a los valores medios son muy grandes. Estos rangos tan amplios son debidos a la cantidad de instancias que dan su resultado en el instante 900 o alrededor.

4.2.4.2 CPU Time con respecto al número de máquinas

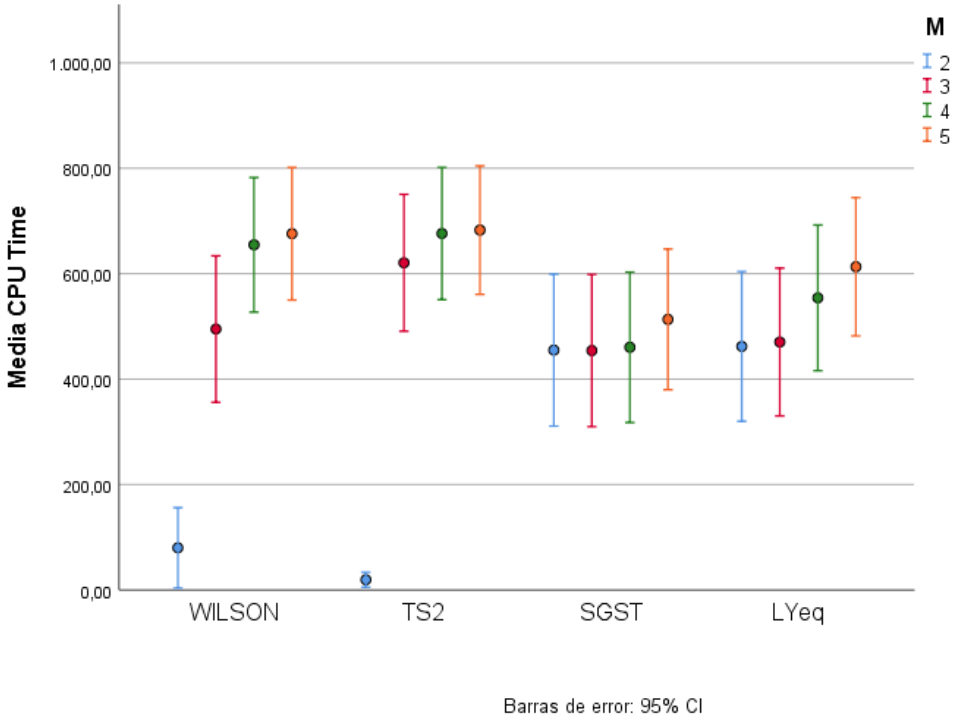
En la *Tabla 19* se observan los valores de la media de CPU Time, junto a la desviación típica de sus valores, divididos según modelos y para cada número de máquinas: 2, 3, 4 y 5 respectivamente. Además, se ha representado gráficamente en la *Gráfica 8*.

M		WILSON	TS2	SGST	LYeq
2	**Media**	79,8147	19,3633	454,9798	462,0225
	N	40	40	40	40
	Desviación	238,66141	44,02092	450,89506	444,31054
3	**Media**	495,0053	620,7228	454,1293	470,339
	N	40	40	40	40
	Desviación	434,53043	406,30174	451,61651	438,36862
4	**Media**	654,7808	676,2658	460,3773	554,0993
	N	40	40	40	40
	Desviación	399,18521	392,57657	446,29861	432,15447
5	**Media**	675,8857	682,8073	513,295	613,1677
	N	40	40	40	40
	Desviación	393,23076	381,19387	417,344	410,1911

Tabla 19. Resultados obtenidos de la media CPU Time con respecto al modelo y M

En cuanto a las ideas extraídas de estos resultados:

- El menor valor medio de CPU Time se da para problemas pequeños, de solo 2 máquinas, resueltos con los modelos de la familia de Wagner.
- A partir de 3 máquinas, para los modelos WILSON y TS2, los valores medios de tiempo de cómputo se disparan.
- Para la familia de modelos de Manne, los resultados de todos los casos son bastante parecidos. Además, los valores medios son menores que los modelos de Wagner. Entre ellos prácticamente se solapan, aunque se puede apreciar el modelo SGST como algo más rápido que el LYeq.



Gráfica 8. Medias del CPU Time con respecto a los modelos y agrupadas según M

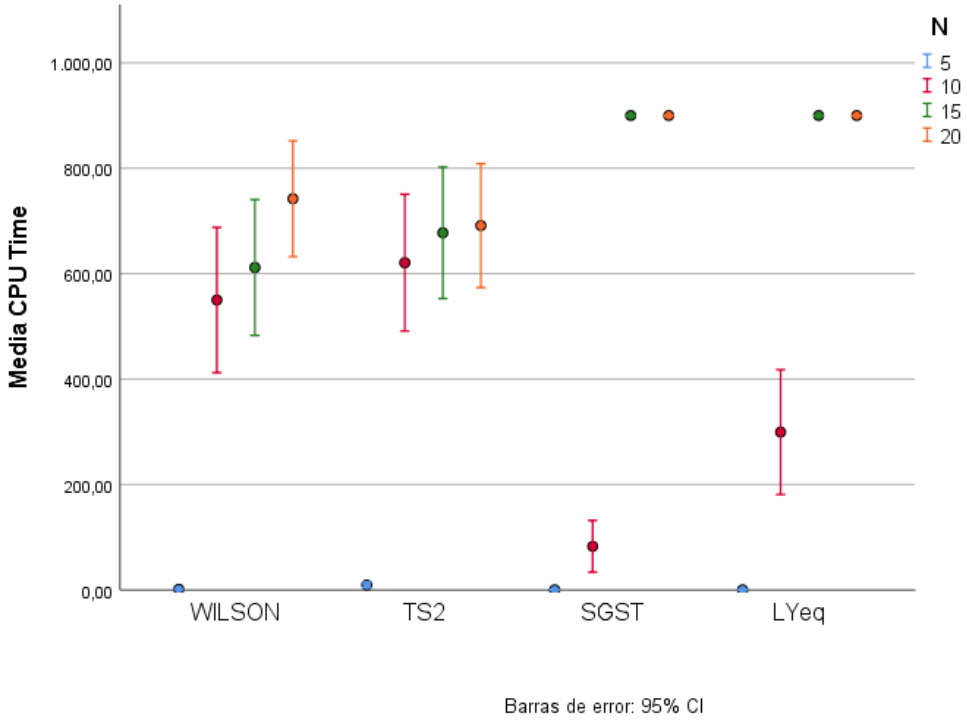
4.2.4.3 CPU Time con respecto al número de trabajos

Para terminar, se procede de igual forma con los siguientes resultados: se exponen en la *Tabla 20* los valores desglosados por modelo y por número de trabajos (5, 10, 15 y 20)

En cuanto a la representación gráfica de dichos valores, se puede encontrar en la *Gráfica 9* y se muestra con el código de colores para diferenciar cada agrupación de instancias según el número de trabajos.

	N	WILSON	TS2	SGST	LYeq
5	**Media**	1,3512	9,319	0,0688	0,106
	N	40	40	40	40
	Desviación	1,49369	16,35624	0,03653	0,04634
10	**Media**	550,0448	620,8873	82,6823	299,4833
	N	40	40	40	40
	Desviación	430,80397	406,01692	153,17106	370,5874
15	**Media**	611,7845	677,568	900,017	900,0135
	N	40	40	40	40
	Desviación	403,3979	390,28358	0,01572	0,01642
20	**Media**	742,306	691,3848	900,0132	900,0257
	N	40	40	40	40
	Desviación	343,81828	367,68729	0,01421	0,04012

Tabla 20. Resultados obtenidos de la media CPU Time con respecto al modelo y N



Gráfica 9. Medias del CPU Time con respecto a los modelos y agrupadas según N

Las ideas que se pueden obtener de este análisis son muy similares al anterior apartado:

- Los problemas más simples, aquellos con 5 trabajos, se resuelven en un tiempo casi despreciable. Conforme aumenta la cantidad de trabajos, el tiempo de computo también lo hace.
- En el caso de los modelos de WILSON y TS2, la media de valores y su desviación típica son muy similares y casi coincidentes las barras del intervalo de confianza.
- El modelo SGST podría considerarse perfecto para solucionar problemas con 5 trabajos y con 10. Sin embargo, cuando los trabajos aumentan a 15 y 20, son problemas tan complejos que se para la simulación en el tiempo límite.
- El modelo LYeq se asemeja al SGST exceptuando el valor algo mayor para problemas de 10 trabajos.

5 CONCLUSIÓN

Este proyecto tiene como objetivo diseñar un modelo de programación lineal capaz de resolver el problema de *Flowshop sin permutación* de la forma más eficiente.

Para llevarlo a cabo se ha partido de los modelos definidos para un problema de *Flowshop de permutación*. Se han adaptado en su forma matemática y finalmente se ha programado la definición de los modelos en leguaje C. El paso final habría sido la ejecución mediante el solver de Gurobi para la obtención de los valores de la función objetivo y el tiempo de cómputo.

Dado que el problema propuesto era complejo, uno de los modelos de la literatura ha quedado fuera del estudio de este proyecto. Se trata del modelo WST, de la familia de modelos de Wagner. Este modelo no se ha podido adaptar de manera efectiva con el uso de las mismas variables que el original y sin añadirle más restricciones de las debidas.

Tras el análisis de los resultados, se puede concluir que es el modelo SGST el que mejor funciona para el problema propuesto, en lo que refiere a encontrar la mejor solución. En segundo lugar, se encontraría el modelo LYeq. Por su parte, los modelos de la familia de Wagner muy pocas veces obtienen la mejor solución de entre las encontradas. Aunque no se haya resuelto mediante el modelo WST, se podría englobar dicho modelo ya que pertenece a esa familia también.

Si lo que interesa es hallar una solución cualquiera en el menor tiempo posible utilizando un *MILP*, la conclusión difiere en algunas ideas de la expuesta en el párrafo anterior. Para problemas pequeños, los modelos de la familia de Wagner dan una solución bastante rápido. Pero para los modelos más complejos siguen siendo los modelos pertenecientes a la familia de Manne los que más rápido resuelven el problema.

Estos resultados pueden ser aplicables a escenarios de la vida real. Se pueden observar entornos de tipo taller de flujo regular, por ejemplo, en lugares dedicados a la preparación de comidas, ensambladoras o empresas que se dedican a la fabricación de determinados productos. En definitiva, cuanto más grande se hace el problema, más complicada se hace la asignación de trabajos a las máquinas y más se tarda en hacer. Este ejemplo se ve fácilmente en un ejemplo de la vida cotidiana, no será lo mismo preparar las comidas de una semana para una familia en un día, que planificar durante un mes las comidas y preparaciones de un restaurante.

REFERENCIAS

Framinan, J. M., Leisten, R. and Ruiz García, R. (2014) *Manufacturing Scheduling Systems*. Springer.

Graham, R. L. *et al.* (1979) 'Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey', *Annals of Discrete Mathematics*. Elsevier.

Optimization, G. (2018) 'Gurobi optimizer reference manual'.

Pérez, P., Fernández-Viagas, V. and Framinan, J. M. (2017) *Programación de la producción*. Apuntes de la asignatura de Programación de Operaciones.

Pinedo, M. L. (2012) *Scheduling*. Springer.

Stafford, E. F., Tseng, F. T. and Gupta, J. N. D. (2005) 'Comparative evaluation of MILP flowshop models', *Journal of the Operational Research Society*, 56, 88–101.

Anexo: Códigos de C

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
   //Definición de variables
   int lm=3;
   int ln=3;
   int iter;
   int longfichero=0;
   int i, j, k, l, r, ch;
   int x1, x2;
   int M, N;
   int T[5][20];
   int P;

   char cadenanombre[150];

   char nombreWILSON[200];
   char nombreTS2[200];
   char nombreSGST[200];
   char nombreLY[200];

   int trabajo [] = {5, 10, 15, 20};
   int maquina [] = {2, 3, 4, 5};

   char cadena [] = "Z";
   char tini [] = "B";
   char tfin [] = "E";
```

```
char tcomp [] = "C";
char dicot [] = "D";
char nuev [] = "Q";
char obj [] = "max";
//Creación lista de nombres(lista_nombres.txt) de la batería de archivos
FILE *nombres;
nombres = fopen("lista_nombres.txt","wt");
for (j=0; j<=ln; j++) {
   for (i=0; i<=lm; i++) {
      for (iter=1; iter<=10; iter++) {
         fprintf (nombres,"inst_%d_%d_p_%d \n", trabajo[j], maquina[i], iter);
      }
   }
}
fclose (nombres);
//Creación lista de nombres modelo WILSON (lista_WILSON.txt) de la batería de archivos
FILE *nombresWILSON;
nombresWILSON = fopen ("lista_WILSON.txt","wt");
for (j=0; j<=ln; j++) {
   for (i=0; i<=lm; i++) {
      for (iter=1; iter<=10; iter++) {
         fprintf (nombresWILSON, "WILSON_inst_%d_%d_p_%d.lp \n", trabajo[j], maquina[i], iter);
      }
   }
}
fclose (nombresWILSON);
//Creación lista de nombres modelo TS2 (lista_TS2.txt) de la batería de archivos
FILE *nombresTS2;
nombresTS2 = fopen ("lista_TS2.txt","wt");
for (j=0; j<=ln; j++) {
   for (i=0; i<=lm; i++) {
      for (iter=1; iter<=10; iter++) {
         fprintf (nombresTS2,"TS2_inst_%d_%d_p_%d.lp \n", trabajo[j], maquina[i], iter);
      }
   }
}
fclose (nombresTS2);
//Creación lista de nombres modelo SGST (lista_SGST.txt) de la batería de archivos
```

```
FILE *nombresSGST;
   nombresSGST = fopen ("lista_SGST.txt","wt");
   for (j=0; j<=ln; j++) {
      for (i=0; i<=lm; i++) {
         for (iter=1; iter<=10; iter++) {
            fprintf (nombresSGST,"SGST_inst_%d_%d_p_%d.lp \n", trabajo[j], maquina[i], iter);
         }
      }
   }
   fclose (nombresSGST);
   //Creación lista de nombres modelo LYeq (lista_LYeq.txt) de la batería de archivos
   FILE *nombresLYeq;
   nombresLYeq = fopen ("lista_LYeq.txt","wt");
   for (j=0; j<=ln;j ++) {
      for (i=0; i<=lm; i++) {
         for (iter=1; iter<=10; iter++) {
            fprintf (nombresLYeq,"LYeq_inst_%d_%d_p_%d.lp \n", trabajo[j], maquina[i], iter);
         }
      }
   }
   fclose(nombresLYeq);
   //Lectura del archivo para contabilizar la longitud del fichero de los nombres
   nombres = fopen ("lista_nombres.txt","rt");
   while ((ch = fgetc (nombres)) != EOF) {
      if (ch == '\n') {
         longfichero++;
      }
   }
   fclose (nombres);
   //Lectura de cada nombre de la lista de nombres y apertura de cada uno de los archivos en cuestión
   nombres = fopen ("lista_nombres.txt","rt"); //abrir en modo lectura el fichero lista_nombres.txt
   nombresWILSON = fopen ("lista_WILSON.txt","rt"); //abrir en modo lectura el fichero lista_WILSON.txt
   nombresTS2 = fopen ("lista_TS2.txt","rt"); //abrir en modo lectura el fichero lista_TS2.txt
   nombresSGST = fopen ("lista_SGST.txt","rt"); //abrir en modo lectura el fichero lista_SGST.txt
   nombresLYeq = fopen ("lista_LYeq.txt","rt"); //abrir en modo lectura el fichero lista_LYeq.txt
   FILE *gurobiWILSON;
   FILE *gurobiTS2;
   FILE *gurobiSGST;
```

```
FILE *gurobiLY;
gurobiWILSON = fopen ("FicheroWILSON.txt","wt");
gurobiTS2 = fopen ("FicheroTS2.txt","wt");
gurobiSGST = fopen ("FicheroSGST.txt","wt");
gurobiLY = fopen ("FicheroLY.txt","wt");
FILE *archivo;
FILE *modelo_WILSON;
FILE *modelo_TS2;
FILE *modelo_SGST;
FILE *modelo_LYeq;
For (k=1; k<=longfichero; k++) {
    fscanf (nombres, "%s\n", cadenanombre); //leer cada línea del fichero lista_nombres.txt y asigno cada nombre a la variable cadenanombre
    fscanf (nombresWILSON, "%s\n", nombreWILSON); //leer cada línea del fichero lista_WILSON.txt y asigno cada nombre a la variable nombreWILSON
    fscanf (nombresTS2, "%s\n", nombreTS2); //leer cada línea del fichero lista_TS2.txt y asigno cada nombre a la variable nombreTS2
    fscanf (nombresSGST, "%s\n", nombreSGST); //leer cada línea del fichero lista_SGST.txt y asigno cada nombre a la variable nombreSGST
    fscanf (nombresLYeq, "%s\n", nombreLY); //leer cada línea del fichero lista_LYeq.txt y asigno cada nombre a la variable nombreLY
    archivo = fopen (cadenanombre, "rt"); //abrir en modo lectura cada fichero con el nombre asignado a la variable cadena nombre
    fscanf (archivo, "%d \n", &M); //leer la primera línea del fichero y asignarla a la variable M
    fscanf (archivo, "%d \n", &N); //leer la segunda línea del fichero y asignarla a la variable N
    for (j=0; j<N; j++) {
      for (i=0; i<M; i++) {
          fscanf (archivo, "%d ", &T[i][j]); //leer los tiempos de proceso del fichero y asignarlos a Tij
      }
      fscanf (archivo, "\n");
    }
    P = 100*N;
//Aplicación del modelo WILSON:
    modelo_WILSON = fopen (nombreWILSON, "wt");
    //Función objetivo
    fprintf (modelo_WILSON, "Minimize \n");
    fprintf (modelo_WILSON, "%s%d%d ", tini, M, N);
      for (i=1; i<=N; i++) {
        fprintf (modelo_WILSON, "+ %d %s%d%d%d ", T[M-1][i-1], cadena, i, N, M);
      }
```

```
//Restricciones
fprintf (modelo_WILSON, "\nSubject To \n");
for (r=1; r<=M; r++) {
   for (i=1; i<=N; i++) {
      for (j=1; j<N; j++) {
         fprintf (modelo_WILSON, "%s%d%d%d + ", cadena, i, j, r);
      }
      if (j=N) {
            fprintf (modelo_WILSON, "%s%d%d%d ", cadena, i, j, r);
         }
      fprintf (modelo_WILSON, "= 1\n");
   }
}
for (r=1; r<=M; r++) {
   for (j=1; j<=N; j++) {
      for (i=1; i<N; i++) {
         fprintf (modelo_WILSON, "%s%d%d%d + ", cadena, i, j, r);
      }
      if (i=N) {
            fprintf (modelo_WILSON, "%s%d%d%d ", cadena, i, j, r);
         }
      fprintf (modelo_WILSON, "= 1\n");
   }
}
for (j=1; j<N; j++) {
   fprintf (modelo_WILSON, "%s%d%d ", tini, 1, j);
      for (i=1; i<=N; i++) {
         fprintf (modelo_WILSON, "+ %d %s%d%d%d ", T[0][i-1], cadena, i, j, 1);
      }
   fprintf(modelo_WILSON, "- %s%d%d = 0\n", tini,1,j+1);
}
fprintf (modelo_WILSON, "%s%d%d = 0\n", tini, 1, 1);
for (r=1; r<M; r++) {
   fprintf (modelo_WILSON, "%s%d%d ", tini, r, 1);
   for (i=1; i<=N; i++) {
      fprintf (modelo_WILSON, "+ %d %s%d%d%d ", T[r-1][i-1], cadena, i, 1, r);
   }
   fprintf (modelo_WILSON, "- %s%d%d = 0\n", tini, r+1, 1);
```

```
    }
    for (r=1; r<M; r++) {
       for (x1=1; x1<=N; x1++) {
          for (x2=1; x2<=N; x2++) {
             for (i=1; i<=N; i++) {
                fprintf (modelo_WILSON, "%s%d%d - %s%d%d ", tini, r, x1, tini, r+1, x2);
                fprintf (modelo_WILSON, "+ %d %s%d%d%d + %d %s%d%d%d ", P, cadena, i, x1, r, P,
cadena, i, x2, r+1);
                fprintf (modelo_WILSON, "+ %d %s%d%d%d ", T[r-1][i-1], cadena, i, x1, r);
                fprintf (modelo_WILSON, "<= %d\n", 2*P);
             }
          }
       }
    }
    for (r=2; r<=M; r++) {
          for (j=1; j<N; j++) {
                fprintf (modelo_WILSON, "%s%d%d ", tini, r, j);
             for (i=1; i<=N; i++) {
                fprintf (modelo_WILSON, "+ %d %s%d%d%d ", T[r-1][i-1], cadena, i, j, r);
             }
             fprintf (modelo_WILSON, "- %s%d%d <= 0\n", tini, r, j+1);
          }
    }
    //Variables
    fprintf (modelo_WILSON, "Binary \n");
    for (r=1; r<=M; r++) {
       for (i=1; i<=N; i++) {
          for (j=1; j<=N; j++) {
             fprintf (modelo_WILSON, "%s%d%d%d ", cadena, i, j, r);
          }
       }
    }
    fprintf (modelo_WILSON, "\nGeneral \n");
    for (r=1; r<=M; r++) {
       for (j=1; j<=N; j++) {
          fprintf (modelo_WILSON, "%s%d%d ", tini, r, j);
       }
    }
```

```
        fprintf (modelo_WILSON, "\nEnd");
        fclose (modelo_WILSON);
        fprintf (gurobiWILSON, "gurobi_cl TimeLimit=900 C:/Users/Beatriz/…/%s \n", nombreWILSON);
        P = 100*N;
    //Aplicación del modelo TS2:
        modelo_TS2 = fopen (nombreTS2,"wt");
        //Función objetivo
        fprintf (modelo_TS2, "Minimize \n");
        fprintf (modelo_TS2, "%s%d%d", tfin, M, N);
        //Restricciones
        fprintf (modelo_TS2, "\nSubject To \n");
        for (r=1; r<=M; r++) {
           for (i=1; i<=N; i++) {
              for (j=1; j<N; j++) {
                 fprintf (modelo_TS2, "%s%d%d%d + ", cadena, i, j, r);
              }
              if (j=N) {
                    fprintf (modelo_TS2, "%s%d%d%d ", cadena, i, j, r);
                 }
              fprintf (modelo_TS2, "= 1\n");
           }
        }
        for (r=1; r<=M; r++) {
           for (j=1; j<=N; j++) {
              for (i=1; i<N; i++) {
                 fprintf (modelo_TS2, "%s%d%d%d + ", cadena, i, j, r);
              }
              if (i=N) {
                    fprintf (modelo_TS2, "%s%d%d%d ", cadena, i, j, r);
                 }
              fprintf (modelo_TS2, "= 1\n");
           }
        }
        for (r=1; r<=M; r++) {
              for (j=1; j<N; j++) {
                    fprintf (modelo_TS2, "%s%d%d ", tfin, r, j);
                 for (i=1; i<=N; i++) {
                    fprintf (modelo_TS2, "+ %d %s%d%d%d ", T[r-1][i-1], cadena, i, j+1, r);
```

```
            }
            fprintf (modelo_TS2, "- %s%d%d <= 0\n", tfin, r, j+1);
        }
    }
    for (r=1; r<M; r++) {
      for (x1=1; x1<=N; x1++) {
        for (x2=1; x2<=N; x2++) {
          for (i=1; i<=N; i++) {
            fprintf (modelo_TS2, "%s%d%d - %s%d%d ", tfin, r, x1, tfin, r+1, x2);
            fprintf (modelo_TS2, "+ %d %s%d%d%d + %d %s%d%d%d ", P, cadena, i, x1, r, P, cadena, i,
x2, r+1);
            fprintf (modelo_TS2, "+ %d %s%d%d%d ", T[r][i-1], cadena, i, x2, r+1);
            fprintf (modelo_TS2, "<= %d\n", 2*P);
          }
        }
      }
    }
    fprintf (modelo_TS2, "%s%d%d ", tfin, 1, 1);
      for (i=1; i<=N; i++) {
        fprintf (modelo_TS2, "- %d %s%d%d%d ", T[0][i-1], cadena, i, 1, 1);
      }
    fprintf (modelo_TS2, ">= 0\n");
    //Variables
    fprintf (modelo_TS2, "Binary \n");
    for (r=1; r<=M; r++) {
      for (i=1; i<=N; i++) {
        for (j=1; j<=N; j++) {
          fprintf (modelo_TS2, "%s%d%d%d ", cadena, i, j, r);
        }
      }
    }
    fprintf (modelo_TS2, "\nGeneral \n");
    for (r=1; r<=M; r++) {
      for (j=1; j<=N; j++) {
        fprintf (modelo_TS2, "%s%d%d ", tfin, r, j);
      }
    }
    fprintf (modelo_TS2, "\nEnd");
```

```
    fclose (modelo_TS2);
    fprintf (gurobiTS2, "gurobi_cl TimeLimit=900 C:/Users/Beatriz/…/%s \n", nombreTS2);
    P = 100*N;
//Aplicación del modelo SGST:
    modelo_SGST = fopen (nombreSGST,"wt");
    //Función objetivo
    fprintf (modelo_SGST, "Minimize\n");
    fprintf (modelo_SGST, "%s%s\n", tcomp, obj);
    //Restricciones
    fprintf (modelo_SGST, "Subject To \n");
    for (i=1; i<=N; i++) {
      fprintf (modelo_SGST, "%s%d%d >= %d\n", tcomp, 1, i, T[0][i-1]);
    }
    for (r=1; r<M; r++) {
      for (i=1; i<=N; i++) {
        fprintf (modelo_SGST, "%s%d%d - %s%d%d >= %d\n", tcomp, r+1, i, tcomp, r, i, T[r][i-1]);
      }
    }
    for (r=1; r<=M; r++) {
      for (i=1; i<l; i++) {
        for (l=i+1; l<=N; l++) {
          fprintf (modelo_SGST, "%s%d%d - %s%d%d + %d %s%d%d%d >= %d\n", tcomp, r, i, tcomp, r,
l, P, dicot, i, l, r, T[r-1][i-1]);
        }
      }
    }
    for (r=1; r<=M; r++) {
      for (i=1; i<l; i++) {
        for (l=i+1; l<=N; l++) {
          fprintf (modelo_SGST, "%s%d%d - %s%d%d - %d %s%d%d%d >= %d\n", tcomp, r, l, tcomp, r,
i, P, dicot, i, l, r, T[r-1][l-1]-P);
        }
      }
    }
    for (i=1; i<=N; i++) {
      fprintf (modelo_SGST, "%s%s - %s%d%d >= 0\n", tcomp, obj, tcomp, M, i);
    }
    //Variables
    fprintf (modelo_SGST, "General \n");
```

```
    for (r=1; r<=M; r++) {
       for (i=1; i<=N; i++) {
          fprintf (modelo_SGST, "%s%d%d ", tcomp, r, i);
       }
    }
    fprintf (modelo_SGST, "\nBinary \n");
    for (r=1; r<=M; r++) {
       for (i=1; i<N; i++) {
          for (l=i+1; l<=N; l++) {
             fprintf (modelo_SGST, "%s%d%d%d ", dicot, i, l, r);
          }
       }
    }
    fprintf (modelo_SGST, "\nEnd");
    fclose (modelo_SGST);
    fprintf (gurobiSGST, "gurobi_cl TimeLimit=900 C:/Users/Beatriz/…/%s \n", nombreSGST);
    P = 100*N;
//Aplicación del modelo LYeq:
    modelo_LYeq = fopen (nombreLY,"wt");
    //Función objetivo
    fprintf (modelo_LYeq, "Minimize\n");
    fprintf (modelo_LYeq, "%s%s\n", tcomp, obj);
    //Restricciones
    fprintf (modelo_LYeq, "Subject To \n");
    for (i=1; i<=N; i++) {
       fprintf (modelo_LYeq, "%s%d%d >= %d\n", tcomp, 1, i, T[0][i-1]);
    }
    for (r=1; r<M; r++) {
       for (i=1; i<=N; i++) {
          fprintf (modelo_LYeq, "%s%d%d - %s%d%d >= %d\n", tcomp, r+1, i, tcomp, r, i, T[r][i-1]);
       }
    }
    for (i=1; i<=N; i++) {
       fprintf (modelo_LYeq, "%s%s - %s%d%d >= 0\n", tcomp, obj, tcomp, M, i);
    }
    for (r=1; r<=M; r++) {
       for (i=1; i<l; i++) {
          for (l=i+1; l<=N; l++) {
```

```
                fprintf (modelo_LYeq, "%d %s%d%d%d + %s%d%d - %s%d%d - %s%d%d%d = %d\n", P, dicot,
i, l, r, tcomp, r, i, tcomp, r, l, nuev, i, l, r, T[r-1][i-1]);
            }
          }
        }
        for (r=1; r<=M; r++) {
          for (i=1; i<l; i++) {
            for (l=i+1; l<=N; l++) {
              fprintf (modelo_LYeq, "%s%d%d%d <= %d\n", nuev, i, l, r, P-T[r-1][i-1]-T[r-1][l-1]);
            }
          }
        }
        //Variables
        fprintf (modelo_LYeq, "General \n");
        for (r=1; r<=M; r++) {
          for (i=1; i<=N; i++) {
            fprintf (modelo_LYeq, "%s%d%d ", tcomp, r, i);
          }
        }
        fprintf (modelo_LYeq, "\nBinary \n");
        for (r=1; r<=M; r++) {
          for (i=1; i<N; i++) {
            for (l=i+1; l<=N; l++) {
              fprintf (modelo_LYeq, "%s%d%d%d ", dicot, i, l, r);
            }
          }
        }
        fprintf (modelo_LYeq, "\nEnd");
        fclose (modelo_LYeq);
        fprintf (gurobiLY, "gurobi_cl TimeLimit=900 C:/Users/Beatriz/…/%s \n", nombreLY);
    }
}
```